Structure-Preserving Smooth Projective Hashing

Olivier Blazy, Céline Chevalier

Structure-Preserving Smooth Projective Hashing

Olivier Blazy[1] and Céline Chevalier[2]

[1] Université de Limoges, XLim, France
[2] Université Panthéon-Assas, Paris, France

Abstract. Smooth projective hashing has proven to be an extremely useful primitive, in particular when used in conjunction with commitments to provide implicit decommitment. This has lead to applications proven secure in the UC framework, even in presence of an adversary which can do adaptive corruptions, like for example Password Authenticated Key Exchange (PAKE), and 1-out-of-m Oblivious Transfer (OT). However such solutions still lack in efficiency, since they heavily scale on the underlying message length.
Structure-preserving cryptography aims at providing elegant and efficient schemes based on classical assumptions and standard group operations on group elements. Recent trend focuses on constructions of structure-preserving signatures, which require message, signature and verification keys to lie in the base group, while the verification equations only consist of pairing-product equations. Classical constructions of Smooth Projective Hash Function suffer from the same limitation as classical signatures: at least one part of the computation (messages for signature, witnesses for SPHF) is a scalar.
In this work, we introduce and instantiate the concept of Structure-Preserving Smooth Projective Hash Function, and give as applications more efficient instantiations for one-round PAKE and three-round OT, and information retrieval thanks to Anonymous Credentials, all UC-secure against adaptive adversaries.

Keywords. Smooth Projective Hash Functions, Structure Preserving, Oblivious Transfer, Password Authenticated Key Exchange, UC Framework, Credentials.

1 Introduction

Smooth Projective Hash Functions (SPHF) were introduced by Cramer and Shoup [CS02] as a means to design chosen-ciphertext-secure public-key encryption schemes. These hash functions are defined such as their value can be computed in two different ways if the input belongs to a particular subset (the *language*), either using a private hashing key or a public projection key along with a private witness ensuring that the input belongs to the language.

In addition to providing a more intuitive abstraction for their original public-key encryption scheme in [CS98], the notion of SPHF also enables new efficient instantiations of their scheme under different complexity assumptions such as DLin, or more generally $k-$MDDH. Due to its usefulness, the notion of SPHF

was later extended to several interactive contexts. One of the most classical applications is to combine them with commitments in order to provide *implicit* decommitments.

Commitment schemes have become a central tool used in cryptographic protocols. These two-party primitives (between a committer and a receiver) are divided into two phases. First, in the *commit* phase, the committer gives the receiver an analogue of a sealed envelope containing a value m, while later in the *opening* phase, the committer reveals m in such a way that the receiver can verify whether it was indeed m that was contained in the envelope. In many applications, for example password-based authenticated key-exchange, in which the committed value is a password, one wants the opening to be implicit, which means that the committer does not really open its commitment, but rather convinces the receiver that it actually committed to the value it pretended to.

An additional difficulty arises when one wants to prove the protocols in the universal composability framework proposed in [Can01]. Skipping the details, when the protocol uses commitments, this usually forces those commitments to be simultaneously *extractable* (meaning that a simulator can recover the committed value m thanks to a trapdoor) and *equivocable* (meaning that a simulator can open a commitment to a value m' different from the committed value m thanks to a trapdoor), which is quite a difficult goal to achieve.

Using SPHF with commitments to achieve an implicit decommitment, the language is usually defined on group elements, with projection keys being group elements, and witnesses being scalars. While in several applications, this has already lead to efficient constructions, the fact that witnesses have to be scalars (and in particular in case of commitments, the randomness used to commit) leads to drastic restrictions when trying to build protocols secure against adaptive corruptions in the UC framework.

This is the classical paradigm of protocol design, where generic primitives used in a modular approach lead to a simple design but quite inefficient constructions, while when trying to move to ad-hoc constructions, the conceptual simplicity is lost and even though efficiency might be gained, a proper security proof gets trickier. Following the same kind of reasoning, [AFG$^+$10] introduced the concept of structure-preserving signatures in order to take the best of both worlds. There has been an ongoing series of work surrounding this notion, for instance [AGHO11, ACD$^+$12, ADK$^+$13, AGOT14a, AGOT14b]. This has shown that structure-preserving cryptography indeed provides the tools needed to have simultaneously simple and efficient protocols.

1.1 Related Work

Smooth Projective Hash Functions (SPHF) were introduced by Cramer and Shoup [CS02] and have been widely used since then, for instance for password-authenticated key exchange (PAKE) [GL03, ACP09, KV09, KV11, BBC$^+$13b], or oblivious transfer (OT) [Kal05, CKWZ13, ABB$^+$13], and a classification was introduced separating SPHF into three main kinds, KV-SPHF,CS-SPHF,GL-SPHF

depending on how the projection keys are generated and when, the former allowing one-round protocols, while the latter have more efficient communication costs (see Section 2.2).

Password-Authenticated Key Exchange (PAKE) protocols were proposed in 1992 by Bellovin and Merritt [BM92] where authentication is done using a simple password, possibly drawn from a small entropy space subject to exhaustive search. Since then, many schemes have been proposed and studied. SPHF have been extensively used, starting with the work of Gennaro and Lindell [GL03] which generalized an earlier construction by Katz, Ostrovsky, and Yung [KOY01], and followed by several other works [CHK+05, ACP09]. More recently, a variant of SPHF proposed by Katz and Vaikuntanathan even allowed the construction of one-round PAKE schemes [KV11, BBC+13b]. The most efficient PAKE scheme so far (using completely different techniques) is the recent Asiacrypt paper [JR14].

The first ideal functionality for PAKE protocols in the UC framework [Can01, CK02] was proposed by Canetti *et al.* [CHK+05], who showed how a simple variant of the Gennaro-Lindell methodology [GL03] could lead to a secure protocol. Though quite efficient, their protocol was not known to be secure against adaptive adversaries, that are capable of corrupting players at any time, and learn their internal states. The first ones to propose an adaptively secure PAKE in the UC framework were Barak *et al.* [BCL+05] using general techniques from multi-party computation. Though conceptually simple, their solution results in quite inefficient schemes.

Recent adaptively secure PAKE were proposed by Abdalla *et al.* [ACP09, ABB+13], following the Gennaro-Lindell methodology with variation of the Canetti-Fischlin commitment [CF01]. However their communication size is growing in the size of the passwords, which is leaking information about an upper-bound on the password used in each exchange.

Oblivious Transfer (OT) was introduced in 1981 by Rabin [Rab81] as a way to allow a receiver to get exactly one out of k messages sent by another party, the sender. In these schemes, the receiver should be oblivious to the other values, and the sender should be oblivious to which value was received. Since then, several instantiations and optimizations of such protocols have appeared in the literature, including proposals in the UC framework [NP01, CLOS02].

More recently, new instantiations have been proposed, trying to reach round-optimality [HK07], or low communication costs [PVW08]. The 1-out-of-2 OT scheme by Choi *et al.* [CKWZ13] based on the DDH assumption seems to be the most efficient one among those that are secure against adaptive corruptions in the CRS model with erasures. But it does not scale to 1-out-of-m OT, for $m > 2$. [ABB+13,BC15] proposed a generic construction of 1-out-of-m OT secure against adaptive corruptions in the CRS model, however the commitment was still growing in the logarithm of the database length. While this is not so much a security issue for OT as this length is supposed to be fixed at the start of the protocol, this is however a weak spot for the efficiency of the final construction.

1.2 Our contributions

Similarly to structure-preserving signatures requiring the message, the signature, and the public keys to be group elements, we propose in this paper the notion of structure-preserving Smooth Projective Hash Functions (SP-SPHF), where both words, witnesses and projection keys are group elements, and hash and projective hash computations are doable with simple pairing-product equations in the context of bilinear groups.

We show how to transform every previously known pairing-less construction of SPHF to fit this methodology, and then propose several applications in which storing a group element as a witness allows to avoid the drastic restrictions that arise when building protocols secure against adaptive corruptions in the UC framework with a scalar as witness. Asking the witness to be a group element enables us to gain more freedom in the simulation (the discrete logarithm of this element and / or real extraction from a commitment). For instance, the simulator can always commit honestly to a random message, since it only needs to modify its witness in the equivocation phase. Furthermore, it allows to avoid bit-per-bit construction.

As an example, we show that the UC-commitment from [FLM11] (while not fitting with the methodology of traditional SPHF from [ABB$^+$13]), is compatible with SP-SPHF and can be used to build UC protocols. As a side contribution, we first generalize this commitment from DLin to the $k-$MDDH assumption from [EHK$^+$13]. The combination of this commitment and the associated SP-SPHF then enables us to give three interesting applications.

Adaptively secure 1-out-of-m Oblivious Transfer. First, we provide a construction of a three-round UC-secure 1-out-of-m OT. Assuming reliable erasures and a single global CRS, we show in Section 5 that our instantiation is UC-secure against adaptive adversaries. Besides having a lesser number of rounds than most recent existing OT schemes with similar security levels, our resulting protocol also has a better communication complexity than the best known solutions so far [CKWZ13, ABB$^+$13] (see Table 1 for a comparison). For ease of readability, we emphasize in this table the SXDH communication cost[1], which is simply $k-$MDDH for $k=1$. Our protocol is "nearly optimal" in the sense that it is still linear in the number of lines m, but the constant in front of m is 1.

Table 1. Comparison with existing UC-secure OT schemes

	Flow	Communication Complexity	Assumption	1-out-of
[CKWZ13]	4	$26\ \mathbb{G} + 7\ \mathbb{Z}_p$	DDH	2
[ABB$^+$13]	3	$(m + 8\log m) \times \mathbb{G}_1 + \log m \times \mathbb{G}_2 + 1 \times \mathbb{Z}_p$	SXDH	m
This paper	3	$(k + 3 + m) \times \mathbb{G}_1 + (k+1) \times \mathbb{G}_2 + 1 \times \mathbb{Z}_p$	$k-$MDDH	m
This paper	3	$(m+4) \times \mathbb{G}_1 + 2 \times \mathbb{G}_2 + 1 \times \mathbb{Z}_p$	SXDH	m

[1] Our OT and PAKE protocols are described in $k-$MDDH but one directly obtains the SXDH versions by simply letting $k=1$ in the commitment presented in Section 4.2.

One-round adaptively secure PAKE**.** Then, we provide an instantiation of a one-round UC-secure PAKE under any $k-$MDDH assumption. Once again, we show in Section 6 that the UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS. Contrarily to most existing one-round adaptively secure PAKE, we show that our scheme enjoys a much better communication complexity while not leaking information about the length of the password used (see Table 2 for a comparison, in particular for the SXDH version). Only [JR14] achieves a slightly better complexity as ours, but only for SXDH, while ours easily extends to $k-$MDDH. Furthermore, our construction is an extension to SP-SPHF of well-known classical constructions based on SPHF, which makes it simpler to understand. We omit [BC15] from the following table, as its contribution is to widen the construction to non-pairing based hypotheses.

Anonymous Credential-Based Message Transmission. Typical credential use involves three main parties. Users need to interact with some authorities to obtain their credentials (assumed to be a set of attributes validated / signed), and then prove to a server that a subpart of their attributes verifies an expect policy. We present a constant-size, round-optimal protocol that allow to use a Credential to retrieve a message without revealing the Anonymous Credentials in a UC secure way, by simply building on the technique proposed earlier in the paper.

Table 2. Comparison with existing UC-secure PAKE schemes where $|\text{password}| = m$

	Adaptive	One-round	Communication complexity	Assumption
[ACP09]	yes	no	$2 \times (2m + 22m\mathfrak{K}) \times \mathbb{G} + \text{OTS}$	DDH
[KV11]	no	yes	$\approx 2 \times 70 \times \mathbb{G}$	DLIN
[BBC$^+$13b]	no	yes	$2 \times 6 \times \mathbb{G}_1 + 2 \times 5 \times \mathbb{G}_2$	SXDH
[ABB$^+$13]	yes	yes	$2 \times 10m \times \mathbb{G}_1 + 2 \times m \times \mathbb{G}_2$	SXDH
[JR14]	yes	yes	$4 \times \mathbb{G}_1 + 4 \times \mathbb{G}_2$	SXDH
this paper	yes	yes	$2 \times (k+3+k \times (k+3)) \times \mathbb{G}_1$ $+2 \times (k+1) \times \mathbb{G}_2$	$k-$MDDH
this paper	yes	yes	$2 \times 8 \times \mathbb{G}_1 + 2 \times 2 \times \mathbb{G}_2$	SXDH

2 Definitions

2.1 Notations

If $\boldsymbol{x} \in \mathcal{S}^n$, then $|\boldsymbol{x}|$ denotes the length n of the vector, and by default vectors are assumed to be column vectors. Further, $x \xleftarrow{\$} \mathcal{S}$ denotes the process of sampling an element x from the set $\mathcal{S}$ uniformly at random.

2.2 Primitives

Encryption. An encryption scheme $\mathcal{C}$ is described through four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$). The formal definitions are given in Appendix A.1.

Commitments. We refer the reader to [ABB+13] for formal definitions and results but we give here an informal overview to help the unfamiliar reader with the following. A *non-interactive labelled commitment scheme* $\mathcal{C}$ is defined by three algorithms:

- $\mathsf{SetupCom}(1^{\mathfrak{K}})$ takes as input the security parameter $\mathfrak{K}$ and outputs the global parameters, passed through the CRS ρ to all other algorithms;
- $\mathsf{Com}^{\ell}(x)$ takes as input a label ℓ and a message x, and outputs a pair (C, δ), where C is the commitment of x for the label ℓ, and δ is the corresponding opening data (a.k.a. decommitment information). This is a probabilistic algorithm.
- $\mathsf{VerCom}^{\ell}(C, x, \delta)$ takes as input a commitment C, a label ℓ, a message x, and the opening data δ and outputs 1 (true) if δ is a valid opening data for C, x and ℓ. It always outputs 0 (false) on $x = \perp$.

The basic properties required for commitments are *correctness* (for all correctly generated CRS ρ, all commitments and opening data honestly generated pass the verification VerCom test), the *hiding property* (the commitment does not leak any information about the committed value) and the *binding property* (no adversary can open a commitment in two different ways). More complex properties (equivocability and extractability) are required by the UC framework and described in Appendix A.2 for lack of space.

Smooth Projective Hash Functions Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup [CS02] for constructing encryption schemes. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. The notion of SPHF has already found numerous applications in various contexts in cryptography (*e.g.* [GL03, Kal05, ACP09, BPV12]).

Definition 1 (Smooth Projective Hashing System). *A Smooth Projective Hash Function over a language $\mathfrak{L} \subset X$, is defined by five algorithms* (Setup, $\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash}$)*:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$ *generates the global parameters* param *of the scheme, and the description of an $\mathcal{NP}$ language $\mathfrak{L}$*
- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$*, outputs a hashing key* hk *for the language $\mathfrak{L}$;*
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$*, derives the projection key* hp*, thanks to the hashing key* hk*,*
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$*, outputs a hash value v, thanks to the hashing key* hk*, and W,*

– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$, *outputs the hash value* v'*, thanks to* hp *and the witness* w *that* $W \in \mathfrak{L}$.

In the following, we consider $\mathfrak{L}$ as a hard-partitioned subset of X, *i.e.* it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element in $X \setminus \mathfrak{L}$.

A Smooth Projective Hash Function SPHF should satisfy the following properties:

– *Correctness*: Let $W \in \mathfrak{L}$ and w a witness of this membership. Then, for all hashing keys hk and associated projection keys hp we have $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$.

– *Smoothness*: For all $W \in X \setminus \mathfrak{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), \\ v = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \in \mathbb{G} \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), v \xleftarrow{\$} \mathbb{G} \end{array} \right\}.$$

A third property called *Pseudo-Randomness*, is implied by the Smoothness on Hard Subset membership languages. If $W \in \mathfrak{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable: for any adversary $\mathcal{A}$ within reasonable time the following advantage is negligible

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SPHF},\mathcal{A}}(\mathfrak{K}) = |\Pr_{\Delta_1}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1]|$$

In [BBC+13b], the authors introduced a new notation for SPHF: for a language $\mathfrak{L}$, there exist a function Γ and a family of functions Θ, such that $\boldsymbol{u} \in \mathfrak{L}$, if and only if, $\Theta(\boldsymbol{u})$ is a linear combination $\boldsymbol{\lambda}$ of the rows of $\Gamma(\boldsymbol{u})$. We furthermore require that a user, who knows a witness of the membership $\boldsymbol{u} \in \mathfrak{L}$, can efficiently compute the linear combination $\boldsymbol{\lambda}$. The SPHF can now then be described as:

– $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, outputs a hashing key $\mathsf{hk} = \boldsymbol{\alpha}$ for the language $\mathfrak{L}$,
– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \boldsymbol{u})$, derives the projection key $\mathsf{hp} = \boldsymbol{\gamma}(\boldsymbol{u})$,
– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \boldsymbol{u})$, outputs a hash value $H = \Theta(\boldsymbol{u}) \odot \boldsymbol{\alpha}$,
– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), \boldsymbol{u}, \boldsymbol{\lambda})$, outputs the hash value $H' = \boldsymbol{\lambda} \odot \boldsymbol{\gamma}(\boldsymbol{u})$.

In the special case where $\mathsf{hp} = \boldsymbol{\gamma}(\boldsymbol{u}) = \boldsymbol{\gamma}$, we speak about $\mathsf{KV\text{-}SPHF}$ when the projection key can be given before seeing the word $\boldsymbol{u}$, and of $\mathsf{CS\text{-}SPHF}$, when the projection key while independent of the word is given after seeing it. (In reference to [KV11, CS02] where those kinds of SPHF were first use). We give in Section 3.3 an exemple of $\mathsf{KV\text{-}SPHF}$ for Cramer-Shoup encryption, both in classical and new notations.

We will need a third property for our one-round PAKE protocol. This property, called strong pseudo-randomness in [BBC+13b], is recalled in Appendix A.3 for lack of space.

2.3 Building Blocks

Decisional Diffie-Hellman (DDH) The Decisional Diffie-Hellman hypothesis says that in a multiplicative group $(p, \mathbb{G}, g)$ when we are given $(g^\lambda, g^\mu, g^\psi)$ for unknown random $\lambda, \mu, \psi \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $\psi = \lambda + \mu$.

Pairing groups. Let GGen be a probabilistic polynomial time (PPT) algorithm that on input $1^\mathfrak{K}$ returns a description $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order p for a $\mathfrak{K}$-bit prime p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ is an efficiently computable (non-degenerated) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$.

Matricial Notations. If $\boldsymbol{A} \in \mathbb{Z}_p^{(k+1)\times n}$ is a matrix, then $\overline{\boldsymbol{A}} \in \mathbb{Z}_p^{k\times n}$ denotes the upper matrix of $\boldsymbol{A}$ and $\underline{\boldsymbol{A}} \in \mathbb{Z}_p^{1\times n}$ denotes the last row of $\boldsymbol{A}$. We use classical notations from [GS08] for operations on vectors (. for the dot product and $\odot$ for the product component-wise). Concatenation of matrices having the same number of lines will be denoted by $\boldsymbol{A}||\boldsymbol{B}$ (where $a||b + c$ should be implicitly parsed as $a||(b + c)$).

We use implicit representation of group elements as introduced in [EHK+13]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$ define $[a]_s = g_s^a \in \mathbb{G}_s$ as the *implicit representation* of a in $\mathbb{G}_s$ (we use $[a] = g^a \in \mathbb{G}$ if we consider a unique group). More generally, for a matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{Z}_p^{n\times m}$ we define $[\boldsymbol{A}]_s$ as the implicit representation of $\boldsymbol{A}$ in $\mathbb{G}_s$:

$$[\boldsymbol{A}]_s := \begin{pmatrix} g_s^{a_{11}} & \dots & g_s^{a_{1m}} \\ & & \\ g_s^{a_{n1}} & \dots & g_s^{a_{nm}} \end{pmatrix} \in \mathbb{G}_s^{n\times m}$$

We will always use this implicit notation of elements in $\mathbb{G}_s$, i.e., we let $[a]_s \in \mathbb{G}_s$ be an element in $\mathbb{G}_s$. Note that from $[a]_s \in \mathbb{G}_s$ it is generally hard to compute the value a (discrete logarithm problem in $\mathbb{G}_s$). Further, from $[b]_T \in \mathbb{G}_T$ it is hard to compute the value $[b]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$ (pairing inversion problem). Obviously, given $[a]_s \in \mathbb{G}_s$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[ax]_s \in \mathbb{G}_s$. Further, given $[a]_1, [b]_2$ one can efficiently compute $[ab]_T$ using the pairing e. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^k$ define $e([\boldsymbol{a}]_1, [\boldsymbol{b}]_2) := [\boldsymbol{a}^\top \boldsymbol{b}]_T \in \mathbb{G}_T$.

If $a \in \mathbb{Z}_p$, we define the $(k + 1)$-vector: $\iota_s(a) := (1_s, \dots, 1_s, [a]_s)$ (this notion can be implicitly extended to vectors $a \in \mathbb{Z}_p^n$), and the $k + 1$ by $k + 1$ matrix $\iota_T(a) := \begin{pmatrix} 1 & \dots & 1 \\ \vdots & \ddots & 1 \\ 1 & 1 & a \end{pmatrix}$.

Assumptions. We recall the definition of the matrix Diffie-Hellman (MDDH) assumption [EHK+13].

Definition 2 (Matrix Distribution). *Let $k \in \mathbb{N}$. We call $\mathcal{D}_k$ a matrix distribution if it outputs matrices in $\mathbb{Z}_p^{(k+1)\times k}$ of full rank k in polynomial time.*

Without loss of generality, we assume the first k rows of $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$ form an invertible matrix. The $\mathcal{D}_k$-Matrix Diffie-Hellman problem is to distinguish the two distributions $([\mathbf{A}], [\mathbf{A}\boldsymbol{w}])$ and $([\mathbf{A}], [\boldsymbol{u}])$ where $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^k$ and $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

Definition 3 ($\mathcal{D}_k$-Matrix Diffie-Hellman Assumption $\mathcal{D}_k$-MDDH). *Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2, T\}$. We say that the $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-MDDH) Assumption holds relative to* GGen *in group $\mathbb{G}_s$ if for all PPT adversaries $\mathcal{D}$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) := |\Pr[\mathcal{D}(\mathcal{G}, [\boldsymbol{A}]_s, [\boldsymbol{A}\boldsymbol{w}]_s) = 1] - \Pr[\mathcal{D}(\mathcal{G}, [\boldsymbol{A}]_s, [\boldsymbol{u}]_s) = 1]| = \mathsf{negl}(\lambda),$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

For each $k \geq 1$, [EHK+13] specifies distributions $\mathcal{L}_k$, $\mathcal{U}_k$, ... such that the corresponding $\mathcal{D}_k$-MDDH assumption is the k-Linear assumption, the k-uniform and others. All assumptions are generically secure in bilinear groups and form a hierarchy of increasingly weaker assumptions. The distributions are exemplified for $k = 2$, where $a_1, \ldots, a_6 \xleftarrow{\$} \mathbb{Z}_p$.

$$\mathcal{L}_2 : \boldsymbol{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} \quad \mathcal{U}_2 : \boldsymbol{A} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \\ a_5 & a_6 \end{pmatrix}.$$

It was also shown in [EHK+13] that $\mathcal{U}_k$-MDDH is implied by all other $\mathcal{D}_k$-MDDH assumptions. In the following, we write $k - \mathsf{MDDH}$ for $\mathcal{D}_k - \mathsf{MDDH}$.

Lemma 4 (Random self reducibility [EHK+13]). *For any matrix distribution $\mathcal{D}_k$, $\mathcal{D}_k$-MDDH is random self-reducible. In particular, for any $m \geq 1$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) + \tfrac{1}{q-1} \geq \mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}')$$

where $\mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}') := \Pr[\mathcal{D}'(\mathcal{G}, [\boldsymbol{A}], [\boldsymbol{A}\boldsymbol{W}]) \Rightarrow 1] - \Pr[\mathcal{D}'(\mathcal{G}, [\boldsymbol{A}], [\boldsymbol{U}]) \Rightarrow 1]$, with $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$, $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k \times m}, \boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{(k+1) \times m}$.

Remark: It should be noted that $\mathcal{L}_1, \mathcal{L}_2$ are respectively the SXDH and DLin assumptions that we recall below for completeness.

Definition 5 (Decisional Linear (DLin [BBS04])). *The Decisional Linear hypothesis says that in a multiplicative group $(p, \mathbb{G}, g)$ when we are given $(g^\lambda, g^\mu, g^{\alpha\lambda}, g^{\beta\mu}, g^\psi)$ for unknown random $\alpha, \beta, \lambda, \mu \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $\psi = \alpha + \beta$.*

Definition 6 (Symmetric External Diffie Hellman (SXDH [ACHdM05])). *This variant of* DDH*, used mostly in bilinear groups in which no computationally efficient homomorphism exists from $\mathbb{G}_2$ in $\mathbb{G}_1$ or $\mathbb{G}_1$ to $\mathbb{G}_2$, states that* DDH *is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Labelled Cramer-Shoup Encryption. We present here the well-known encryption schemes based on DDH, and we show in Section 4 how to extend it to $\mathcal{D}_k - \mathsf{MDDH}$. We focus on Cramer-Shoup [CS98] in all the following of the paper, but one easily obtains the same results on El Gamal IND-CPA scheme [ElG84]

by simply omitting the corresponding parts. We are going to rely on the IND-CCA property to be able to decrypt queries in the simulation.

Vanilla Cramer-Shoup Encryption. The Cramer-Shoup encryption scheme is an IND-CCA version of the ElGamal Encryption. We present it here as a labeled public-key encryption scheme, the classical version is done with $\ell = \emptyset$.

- $\mathsf{Setup}(1^\mathfrak{K})$ generates a group $\mathbb{G}$ of order p, with a generator g
- $\mathsf{KeyGen}(\mathsf{param})$ generates $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. It also chooses a Collision-Resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r)$, for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/(u_1^z)$ and outputs M. Otherwise, one outputs $\perp$.

The security of the scheme is proven under the DDH assumption and the fact the hash function used is a Universal One-Way Hash Function.

In following work [CS02] they refined the proof, explaining that the scheme can be viewed as a 2-Universal Hash Proof on the language of valid Diffie Hellman tuple.

Vanilla Cramer-Shoup Encryption with Matricial Notations.

- $\mathsf{Setup}(1^\mathfrak{K})$ generates a group $\mathbb{G}$ of order p, with a generator g, with an underlying matrix assumption $\mathcal{D}_1$ using a base matrix $[\boldsymbol{A}] \in \mathbb{G}^{2\times 1}$;
- $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = \boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^2$ (with $\boldsymbol{t}_1 = (x_1, x_2)$, $\boldsymbol{t}_2 = (y_1, y_2)$ and $\boldsymbol{z} = (z, 1)$), and sets $c = \boldsymbol{t}_1\boldsymbol{A}, d = \boldsymbol{t}_2\boldsymbol{A}, h = \boldsymbol{z}\boldsymbol{A}$. It also chooses a hash function $\mathfrak{H}_K$ in a collision-resistant hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = ([A], [c], [d], [h], \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, [m]; r)$, for a message $M = [m] \in \mathbb{G}$ and random scalar $r \xleftarrow{\$} \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = [\boldsymbol{A}r]), e = [\boldsymbol{h}r + m], v = [(\boldsymbol{c} + \boldsymbol{d} \odot \xi)\boldsymbol{r}]$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether v is consistant with $\boldsymbol{t}_1, \boldsymbol{t}_2$.
  If it is, one computes $M = [e - (\boldsymbol{u}\boldsymbol{z})]$ and outputs M. Otherwise, one outputs $\perp$.

Groth-Sahai Proof System. Groth and Sahai [GS08] proposed non-interactive zero-knowledge proofs of satisfiability of certain equations over bilinear groups, called *pairing product equations*. Using as witness group elements (and scalars) which satisfy the equation, the prover starts with making commitments on them. To prove satisfiability of an equation (which is the statement of the proof), a Groth-Sahai proof uses these commitments and shows that the committed values satisfy the equation. The proof consists again of group elements and is verified by a pairing equation derived from the statement.

We refer to [GS08] for details of the Groth-Sahai proof system, and to [EHK+13] for the compatibility with the k-MDDH assumptions. More details can be found in Appendix B of the additional content. We are going to give a rough idea of the technique for SXDH.

To prove that committed variables satisfy a set of relations, the Groth-Sahai techniques require one commitment per variable and one proof element (made of a constant number of group elements) per relation. Such proofs are available for pairing-product relations and for multi-exponentiation equations.

When based on the SXDH assumption, the commitment key is of the form $\mathbf{u}_1 = (u_{1,1}, u_{1,2}), \mathbf{u}_2 = (u_{2,1}, u_{2,2}) \in \mathbb{G}_1^2$ and $\mathbf{v}_1 = (v_{1,1}, v_{1,2}), \mathbf{v}_2 = (v_{2,1}, v_{2,2}) \in \mathbb{G}_2^2$. We write

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix} \quad \text{and} \quad \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{pmatrix}.$$

The Setup algorithm initializes the parameters as follows: $\mathbf{u}_1 = (g_1, u)$ with $u = g_1^\lambda$ and $\mathbf{u}_2 = {\mathbf{u}_1}^\mu$ with $\lambda, \mu \xleftarrow{\$} \mathbb{Z}_p^*$, which means that $\mathbf{u}$ is a Diffie-Hellman tuple in $\mathbb{G}_1$, since $\mathbf{u}_1 = (g_1, g_1^\lambda)$ and $\mathbf{u}_2 = (g_1^\mu, g_1^{\lambda\mu})$. The TSetup algorithm will use instead $\mathbf{u}_2 = {\mathbf{u}_1}^\mu \odot (1, g_1)^{-1}$: $\mathbf{u}_1 = (g_1, g_1^\lambda)$ and $\mathbf{u}_2 = (g_1^\mu, g_1^{\lambda\mu-1})$. And it is the same in $\mathbb{G}_2$ for $\mathbf{v}$. Depending on the definition of $\mathbf{u}_2, \mathbf{v}_2$, this commitment can be either perfectly hiding or perfectly binding. The two parameter initializations are indistinguishable under the SXDH assumption.

To commit to $X \in \mathbb{G}_1$, one chooses randomness $s_1, s_2 \in \mathbb{Z}_p$ and sets $\mathcal{C}(X) = (1, X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2} = (1, X) \odot (u_{1,1}^{s_1}, u_{1,2}^{s_1}) \odot (u_{2,1}^{s_2}, u_{2,2}^{s_2}) = (u_{1,1}^{s_1} \cdot u_{2,1}^{s_2}, X \cdot u_{1,2}^{s_1} \cdot u_{2,2}^{s_2})$. Similarly, one can commit to element in $\mathbb{G}_2$ and scalars in $\mathbb{Z}_p$. The committed group elements can be extracted if $\mathbf{u}_2$ is linearly dependant of $\mathbf{u}_1$ by knowing the discrete logarithm x_1 between $\mathbf{u}_{1,1}$ and $\mathbf{u}_{2,2}$: $c_2/(c_1^{x_1}) = X$.

In the following we are going to focus on proof of linear multi-scalar exponentiation in $\mathbb{G}_1$, that is to say we are going to prove equations of the form $\prod_i A_i^{y_i} = A$ where A_i are public elements in $\mathbb{G}_1$ and y_i are going to be scalars committed into $\mathbb{G}_2$.

2.4 Protocols

UC Framework. The goal of this simulation-based model [Can01] is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. Due to lack of space, a short introduction to the UC framework is given in Appendix C of the additional content.

Oblivious Transfer and Password-Authenticated Key-Exchange. The security properties for these two protocols are given in terms of ideal functionalities in Appendix C of the additional content.

3 Structure-Preserving Smooth Projective Hashing

3.1 Definition

In this section, we are now going to narrow the classical definition of Smooth Projective Hash Functions to what we are going to name Structure-Preserving Smooth Projective Hash Functions, in which both words, witnesses and projection keys are group elements.

Since witnesses now become group elements, this allows a full compatibility with Groth and Sahai methodology [GS08], such that for instance possessing a Non-Interactive Zero-Knowledge Proof of Knowledge can become new witnesses of our SP-SPHF, leading to interesting applications, as described later on.

As we are in the context of Structure Preserving cryptography, we assume the existence of a (prime order) bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, e)$, and consider Languages (sets of elements) $\mathfrak{L}$ defined over this group. The hash space is usually $\mathbb{G}_T$, the projection key space a group $\mathbb{G}_1^m \times \mathbb{G}_2^n$ and the witness space a group $\mathbb{G}_1^n \times \mathbb{G}_2^m$.

Definition 7 (Structure-Preserving Smooth Projective Hash Functions). *A Structure-Preserving Smooth Projective Hash Function over a language $\mathfrak{L} \subset X$ onto a set $\mathcal{H}$, is defined by 4 algorithms* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$*:*

- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$*, outputs a hashing key* hk *for the language* $\mathfrak{L}$*;*
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$*, derives the projection key* hp *thanks to the hashing key* hk*.*
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$*, outputs a hash value $H \in \mathcal{H}$, thanks to the hashing key* hk*, and W*
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$*, outputs the value $H' \in \mathcal{H}$, thanks to* hp *and the witness w that $W \in \mathfrak{L}$.*

3.2 Properties

Properties are then inherited by those of classical Smooth Projective Hash Functions.

- *Correctness*: On honest computations with (W, w) compatible with $\mathfrak{L}$, we have $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w) = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$.
- *Smoothness*: For all $W \in X \setminus \mathfrak{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), \\ v = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \in \mathbb{G}_T \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), v \xleftarrow{\$} \mathbb{G}_T \end{array} \right\}.$$

This is formalized by

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{SPHF}}(\mathfrak{K}) = \sum_{V \in \mathbb{G}} \left| \Pr_{\Delta_1}[v = V] - \Pr_{\Delta_0}[v = V] \right| \text{ is negligible.}$$

As usual, a derivative property called *Pseudo-Randomnness*, says the previous distribution are computationally indistinguishable from words in the language while the witnesses remain unknown. This is implied by the Smoothness on Hard Subset membership languages.

3.3 Retro-compatibility

Constructing SP-SPHF is not that hard of a task. A first naive approach allows to transform every pairing-less SPHF into a SP-SPHF in a bilinear setting. It should be noted that while the resulting Hash/ProjHash values live in the target group, nearly all use cases encourage to use a proper hash function on them before computing anything using their value, hence the communication cost would remain the same. (Only applications where one of the party has to provide an additional proof that the ProjHash was honestly computed might be lost, but besides proof of negativity from [BCV15], this never arises.)

To this goal, simply given a new generator $f \in \mathbb{G}_2$, and a scalar witness vector λ, one generates the new witness vector $\Lambda = [f \odot \boldsymbol{\lambda}]_2$. Words and projection keys belong to $\mathbb{G}_1$, and hash values to $\mathbb{G}_T$. Any SPHF can thus be transformed into an SP-SPHF in the following way:

	SPHF	SP-SPHF
Word $\boldsymbol{u}$	$[\boldsymbol{\lambda} \odot \Gamma(\boldsymbol{u})]_1$	$[\boldsymbol{\lambda} \odot \Gamma(\boldsymbol{u})]_1$
Witness w	$\boldsymbol{\lambda}$	$\boldsymbol{\Lambda} = [f \odot \boldsymbol{\lambda}]_2$
hk	$\boldsymbol{\alpha}$	$\boldsymbol{\alpha}$
$\mathsf{hp} = [\boldsymbol{\gamma}(\boldsymbol{u})]_1$	$[\Gamma(\boldsymbol{u}) \odot \boldsymbol{\alpha}]_1$	$[\Gamma(\boldsymbol{u}) \odot \boldsymbol{\alpha}]_1$
$\mathsf{Hash}(\mathsf{hk}, \boldsymbol{u})$	$[\Theta(\boldsymbol{u}) \odot \boldsymbol{\alpha}]_1$	$[f \odot \Theta(\boldsymbol{u}) \odot \boldsymbol{\alpha}]_T$
$\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{u}, w)$	$[\boldsymbol{\lambda} \odot \boldsymbol{\gamma}(\boldsymbol{u})]_1$	$[\boldsymbol{\Lambda} \odot \boldsymbol{\gamma}(\boldsymbol{u})]_T$

- *Correctness* is inherited for words in $\mathcal{L}$ as this reduces to computing the same values but in $\mathbb{G}_T$.
- *Smoothness*: For words outside the language, the projection keys, remaining unchanged, do not reveal new information, so that the smoothness will remain preserved.
- *Pseudo-Randomness*: Without any witness, words inside the language are indistinguishable from words outside the language (under the subgroup decision assumption), hence the hash values remain pseudo-random.

It should be noted that in case this does not weaken the subgroup decision assumption (k-MDDH in the following) linked to the original language, one can set $\mathbb{G}_1 = \mathbb{G}_2$.

We give in Figure 1 two examples of regular Smooth Projective Hash Functions on Diffie-Hellman and Cramer Shoup encryption of M, where $\alpha = \mathcal{H}(\mathbf{u}, e)$, and their counterparts with SP-SPHF. El Gamal being a simplification of Cramer Shoup, we skip the description of the associated SP-SPHF. We also give in Figure 2 the matricial version of Cramer Shoup encryption, in which we denote by C' the Cramer-Shoup encryption C of M in which we removed M.

	SPHF	SP-SPHF
DH	h^r, g^r	h^r, g^r
Witness w	r	g_2^r
hk	λ, μ	λ, μ
hp	$h^\lambda g^\mu$	$h^\lambda g^\mu$
$\mathsf{Hash}(\mathsf{hk}, \boldsymbol{u})$	$(h^r)^\lambda (g^r)^\mu$	$e((h^r)^\lambda (g^r)^\mu, g_2)$
$\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{u}, w)$	hp^r	$e(\mathsf{hp}, g_2^r)$
CS(M;r)	$h^r M, f^r, g^r, (cd^\alpha)^r$	$h^r M, f^r, g^r, (cd^\alpha)^r$
Witness w	r	g_2^r
hk	$\lambda_1, \lambda_2, \mu, \nu, \eta$	$\lambda_1, \lambda_2, \mu, \nu, \eta$
hp	$h^{\lambda_1} f^\mu g^\nu c^\eta, h^{\lambda_2} d^\nu$	$h^{\lambda_1} f^\mu g^\nu c^\eta, h^{\lambda_2} d^\nu$
$\mathsf{Hash}(\mathsf{hk}, \boldsymbol{u})$	$(h^r)^{\lambda_1+\alpha\lambda_2} (f^r)^\mu (g^r)^\nu ((cd^\alpha)^r)^\mu$	$e((h^r)^{\lambda_1+\alpha\lambda_2} (f^r)^\mu (g^r)^\nu ((cd^\alpha)^r)^\mu, g_2)$
$\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{u}, w)$	$(\mathsf{hp}_1 \mathsf{hp}_2^\alpha)^r$	$e(\mathsf{hp}_1 \mathsf{hp}_2^\alpha, g_2^r)$

Fig. 1. Example of conversion of classical SPHF into SP-SPHF

3.4 Possible Applications

Nearly Constant 1-out-of-m Oblivious Transfer Using FLM. Recent pairing-based constructions [CKWZ13, ABB+13] of Oblivious Transfer use SPHF to mask each line of a database with the hash value of as SPHF on the language corresponding to the first flow being a commitment of the said line.

Sadly, those constructions require special UC commitment on scalars, with equivocation and extraction capacities, leading to very inefficient constructions. In 2011, [FLM11] proposed a UC commitment, whose decommitment operation is done via group elements. In section 5, we are going to show how to combine the existing constructions with this efficient commitment using SP-SPHF, in order to obtain a very efficient round-optimal where there is no longer a growing overhead due to the commitment. As a side result, we show how to generalize the FLM commitment to any MDDH assumption.

Round-Optimal Password Authenticated Key Exchange with Adaptive Corruptions. Recent developments around SPHF-based PAKE have either lead to Round-Optimal PAKE in the BPR model [BPR00], or with static corruptions [KV11, BBC+13b]. In order to achieve round-optimality, [ABB+13] needs to do a bit-per-bit commitment of the password, inducing a communication cost proportional to the maximum password length.

In the following, we show how to take advantage of the SP-SPHF constructed on the FLM commitment to propose a One-Round PAKE UC secure against adaptive adversaries, providing a constant communication cost.

Using a ZKPK as a witness, Anonymous Credentials. Previous applications allow more efficient instantiations of protocols already using scalar-based SPHF. However, one can imagine additional scenarios, where a scalar based approach may not be possible, due to the inherent nature of the witness used.

	SPHF	SP-SPHF
CS(M;r)	$[hr + M, \boldsymbol{A}r, (c + d\alpha)r]$	$[hr + M, \boldsymbol{A}r, (c + d\alpha)r]_1$
$\boldsymbol{B} : \begin{pmatrix} h \\ f \\ g \\ c \end{pmatrix}$	$\left[\boldsymbol{B}r + \begin{pmatrix} 0 \\ 0 \\ 0 \\ d \end{pmatrix} \alpha r + \begin{pmatrix} M \\ 0 \\ 0 \\ 0 \end{pmatrix}\right]$	$\left[\boldsymbol{B}r + \begin{pmatrix} 0 \\ 0 \\ 0 \\ d \end{pmatrix} \alpha r + \begin{pmatrix} M \\ 0 \\ 0 \\ 0 \end{pmatrix}\right]_1$
Witness w	r	$[r]_2$
hk	$\lambda_1, \lambda_2, \mu, \nu, \eta$	$\lambda_1, \lambda_2, \mu, \nu, \eta$
hp	$\left[\begin{pmatrix} \lambda_1 & \mu & \nu & \eta \end{pmatrix} \boldsymbol{B} + \begin{pmatrix} \lambda_2 & 0 & 0 & \eta \end{pmatrix} \begin{pmatrix} h \\ 0 \\ 0 \\ d \end{pmatrix}\right]$	$\left[\begin{pmatrix} \lambda_1 & \mu & \nu & \eta \end{pmatrix} \boldsymbol{B} + \begin{pmatrix} \lambda_2 & 0 & 0 & \eta \end{pmatrix} \begin{pmatrix} h \\ 0 \\ 0 \\ d \end{pmatrix}\right]_1$
$\mathsf{Hash}(\mathsf{hk}, \boldsymbol{u})$	$\left[\begin{pmatrix} \lambda_1 + \alpha\lambda_2 & \mu & \nu & \eta \end{pmatrix} (C')\right]$	$\left[\begin{pmatrix} \lambda_1 + \alpha\lambda_2 & \mu & \nu & \eta \end{pmatrix} (C')\right]_T$
$\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{u}, w)$	$[(\mathsf{hp}_1 + \alpha\mathsf{hp}_2)r]$	$[(\mathsf{hp}_1 + \alpha\mathsf{hp}_2)r]_2$

Fig. 2. Example of conversion of SPHF into SP-SPHF (matricial notations)

For example, one should consider a strong authentication scenario, in which each user possesses an identifier delivered by an authority, and a certification on a commitment to this identifier, together with a proof of knowledge that this commitment is indeed a commitment to this identifier. (Such scenario can be transposed to the delivery of a Social Security Number, where a standalone SSN may not be that useful, but a SSN officially linked to someone is a sensitive information that should be hidden.) In this scenario, a user who wants to access his record on a government service where he is already registered, should give the certificate, and then would use an implicit proof that this corresponds to his identifier. With our technique, the server would neither learn the certificate in the clear nor the user identifier (if he did not possess it earlier), and the user would be able to authenticate only if his certificate is indeed on his committed identifier.

In our scenario, we could even add an additional step, such that Alice does not interact directly with Bob but can instead use a pawn named Carol. She could send to Carol a commitment to the signature on her identity, prove in a black box way that it is a valid signature on an identity, and let Carol do the interaction on her behalf. For example, to allow a medical practitioner to access some subpart of her medical record concerning on ongoing treatment, in this case, Carol would need to anonymously prove to the server that she is indeed a registered medical practitioner, and that Alice has given her access to her data.

4 Encryption and Commitment Schemes Based on k-MDDH

4.1 $k-$MDDH Cramer-Shoup Encryption

In this paper, we supersede the previous constructions with a $k-\mathsf{MDDH}$ based one:

- $\mathsf{Setup}(1^\mathfrak{K})$ generates a group $\mathbb{G}$ of order p, with an underlying matrix assumption using a base matrix $[\boldsymbol{A}] \in \mathbb{G}^{k+1\times k}$;
- $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = \boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, and sets, $\boldsymbol{c} = \boldsymbol{t}_1 \in \mathbb{Z}_p^k, \boldsymbol{d} = \boldsymbol{t}_2\boldsymbol{A} \in \mathbb{Z}_p^k, \boldsymbol{h} = \boldsymbol{z}\overline{\boldsymbol{A}} \in \mathbb{Z}_p^k$. It also chooses a hash function $\mathfrak{H}_K$ in a collision-resistant hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = ([\boldsymbol{c}], [\boldsymbol{d}], [\boldsymbol{h}], [\boldsymbol{A}], \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, [m]; \boldsymbol{r})$, for a message $M = [m] \in \mathbb{G}$ and random scalars $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^k$, the ciphertext is $C = (\mathbf{u} = [\boldsymbol{A}\boldsymbol{r}]), e = [\boldsymbol{h}\boldsymbol{r} + m], v = (\boldsymbol{c} + \boldsymbol{d} \odot \xi)\boldsymbol{r}]_1$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether v is consistent with $\boldsymbol{t}_1, \boldsymbol{t}_2$.
  If it is, one computes $M = [e - (\boldsymbol{u}\boldsymbol{z})]$ and outputs M. Otherwise, one outputs $\perp$.

Theorem 8. *The $k-\mathsf{MDDH}$ Cramer-Shoup Encryption is IND-CCA 2 under $k-\mathsf{MDDH}$ assumption and the collision resistance (universal one-wayness) of the Hash Family.*

Proof. To sketch the proof of the theorem, one should remember that the original proof articulate around three main cases noting $\ell, \mathbf{u}, e, v$ the challenge query, and $\ell', \mathbf{u}, e', v'$ the current decryption query:

- $(\ell, \mathbf{u}, e) = (\ell', \mathbf{u}', e')$ but $v \neq v'$. This will fail as v is computed to be the correct checksum, hence we can directly reject the decryption query.
- $(\ell, \mathbf{u}, e) \neq (\ell', \mathbf{u}', e')$ but $\xi = \xi'$, this is a collision on the Hash Function.
- $(\ell, \mathbf{u}, e, v) \neq (\ell, \mathbf{u}, e, v)$ and $\xi \neq \xi'$. This is the argument revolving around the 2-Universality of the Hash Proof system defined by $\boldsymbol{c}, \boldsymbol{d}$. $\boldsymbol{c}, \boldsymbol{d}$ gives $2k$ equations in $2k+2$ variables, hence answering decryption queries always in the same span can give at most 1 more equation leaving at least 1 degree of freedom in the system. □

Structure-Preserving Smooth Projective Hash Function

For ease of readability we are going to set $\boldsymbol{B} = \begin{bmatrix} \begin{pmatrix} h \\ \boldsymbol{A} \\ \boldsymbol{c} \end{pmatrix} \end{bmatrix}$ and $\boldsymbol{D} = \begin{bmatrix} \begin{pmatrix} 0 \\ \vdots \\ \boldsymbol{d} \end{pmatrix} \end{bmatrix}$, and write $C' = [\boldsymbol{B}\boldsymbol{r} + \xi\boldsymbol{D}\boldsymbol{r}]_1$ the ciphertext without the message M.

- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, $\Lambda \xleftarrow{\$} \mathbb{Z}_p^{k+2}$, $\lambda \xleftarrow{\$} \mathbb{Z}_p$, and sets $\mathsf{hk}_1 = \Lambda, \mathsf{hk}_2 = \begin{pmatrix} \lambda \\ \mathbf{0} \\ \Lambda_{k+2} \end{pmatrix}$;

– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, $\mathsf{hp}_1 = \mathsf{hk}_1^\top \boldsymbol{B}, \mathsf{hp}_2 = \mathsf{hk}_2^\top \begin{pmatrix} h \\ \mathbf{0} \\ d \end{pmatrix}$;
– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, outputs a hash value $H = [(\mathsf{hk}_1 + \xi \mathsf{hk}_2)^\top C']_T$;
– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$, outputs the value $H' = [(\mathsf{hp}_1 + \xi \mathsf{hp}_2)\boldsymbol{r}]_T$.

The Smoothness comes inherently from the fact that we have $2k+2$ unknowns in hk while hp gives at most $2k$ equations. hence an adversary has a negligible chance to find the real values.

4.2 A Universally Composable Commitment with Adaptive Security Based on MDDH

We first show how to simply generalize FLM's commitment from DLin to **k**-MDDH.

FLM's Commitment on DLin. At Asiacrypt 2011, Fischlin, Libert and Manulis presented a universally composable commitment [FLM11] with adaptive security based on the Decision Linear assumption [BBS04]. We show here how to generalize their scheme to the Matrix Decisional Diffie-Hellman assumption from [EHK+13] and recalled in Section 2. We first start by recalling their original scheme. Note that sid denotes the session identifier and cid the commitment identifier and that the combination $(\mathsf{sid}, \mathsf{cid})$ is globally unique, as in [HMQ04, FLM11].

– **CRS Generation:** $\mathsf{SetupCom}(1^{\mathfrak{K}})$ chooses a bilinear group $(p, \mathbb{G}, \mathbb{G}_T)$ of order $p > 2^{\mathfrak{K}}$, a generator g of $\mathbb{G}$, and sets $g_1 = g^{\alpha_1}$ and $g_2 = g^{\alpha_2}$ with random $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$. It defines the vectors $\mathbf{g_1} = (g_1, 1, g)$, $\mathbf{g_2} = (1, g_2, g)$ and $\mathbf{g_3} = \mathbf{g_1}^{\xi_1} \mathbf{g_2}^{\xi_2}$ with random $\xi_1, \xi_2 \in \mathbb{Z}_p^*$, which form a Groth-Sahai CRS $\mathbf{g} = (\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3})$ for the perfect soundness setting. It then chooses a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$ and generates a public key $\mathsf{pk} = (X_1, \ldots, X_6)$ for the linear Cramer-Shoup encryption scheme. The CRS consists of $\mathsf{crs} = (\mathfrak{K}, \mathbb{G}, \mathbb{G}_T, g, \mathbf{g}, H, \mathsf{pk})$.
– **Commitment algorithm:** $\mathsf{Com}(\mathsf{crs}, M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$: to commit to message $M \in \mathbb{G}$ for party P_j, party P_i parses crs as $(\mathfrak{K}, \mathbb{G}, \mathbb{G}_T, g, \mathbf{g}, H, \mathsf{pk})$ and conducts the following steps:
  - It chooses random exponents r, s in $\mathbb{Z}_p$ and computes a linear Cramer-Shoup encryption $\psi_{CS} = (U_1, U_2, U_3, U_4, U_5)$ of $M \in \mathbb{G}$ under the label $\ell = P_i \| \mathsf{sid} \| \mathsf{cid}$ and the public key pk.
  - It generates a NIZK proof $\pi_{val-enc}$ that $\psi_{CS} = (U_1, U_2, U_3, U_4, U_5)$ is indeed a valid encryption of $M \in \mathbb{G}$. This requires to commit to exponents r, s and prove that these exponents satisfy the multi-exponentiation equations $U_1 = {g_1}^r$, $U_2 = {g_2}^s$, $U_3 = g^{r+s}$, $U_4/M = {X_5}^r {X_6}^s$ and $U_5 = ({X_1 {X_3}^\alpha})^r \cdot ({X_2 {X_4}^\alpha})^s$.
  - P_i erases (r, s) after the generation of $\pi_{val-enc}$ but retains the $D_M = \pi_{val-enc}$.

  The commitment is ψ_{CS}.

– **Verification algorithm:** $\mathsf{VerCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$: checks the proof $\pi_{val-enc}$ and ignores the opening if the verification fails.
– **Opening algorithm:** $\mathsf{OpenCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$: reveals M and $D_M = \pi_{val-enc}$ to P_j.

The extraction algorithm uses the Cramer-Shoup decryption algorithm, while the equivocation uses the simulator of the NIZK. It is shown in [ABB+13] that the IND-CCA security notion for C and the computational soundness of π make it strongly-binding-extractable, the IND-CCA security notion and the zero-knowledge property of the NIZK provide the strong-simulation-indistinguishability.

Moving to k-MDDH: We now show how to extend the previous commitment to the $k-\mathsf{MDDH}$ assumption. Compared to the original version of the commitment, we split the proof $\pi_{val-enc}$ into its two parts: the NIZK proof denoted here as $[\boldsymbol{\Pi}]_1$ is still revealed during the opening algorithm, while the Groth-Sahai commitment $[\boldsymbol{R}]_2$ of the randomness $\mathbf{r}$ of the Cramer-Shoup encryption is sent during the commitment phase. Furthermore, since the hash value in the Cramer Shoup encryption is used to link the commitment with the session, we include this value $[\boldsymbol{R}]_2$ to the label, in order to ensure that this extra commitment information given with the ciphertext is the original one. We refer the reader to the original security proof in [FLM11, Theorem 1], which remains exactly the same, since this additional commitment provides no information (either computationally or perfectly, depending on the CRS), and since the commitment $[\boldsymbol{R}]_2$ is not modified in the equivocation step (only the value $[\boldsymbol{\Pi}]_1$ is changed).

– **CRS Generation:** algorithm $\mathsf{SetupCom}(1^{\mathfrak{K}})$ chooses a bilinear asymmetric group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of order $p > 2^{\mathfrak{K}}$, and a set of generators $[\boldsymbol{A}]_1$ corresponding to the underlying matrix assumption.
As explained in [EHK+13], following their notations, one can define a Groth-Sahai CRS by picking $\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, and setting $[\boldsymbol{U}]_2 = [\boldsymbol{B}||\boldsymbol{B}\boldsymbol{w}]_2$ for a binding CRS, and $[\boldsymbol{B}||\boldsymbol{B}\boldsymbol{w} + (0||z)^\top]_2$ otherwise, where $[\boldsymbol{B}]_2$ is an $k-\mathsf{MDDH}$ basis, and $\boldsymbol{w}, z$ are the elements defining the challenge vector.
For the Cramer Shoup like CCA-2 encryption, one additionally picks $\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, and a Universal One-Way Hash Function $\mathcal{H}$ and sets $[\boldsymbol{h}]_1 = [\boldsymbol{z} \cdot \boldsymbol{A}]_1$, $[\boldsymbol{c}]_1 = [\boldsymbol{t}_1\boldsymbol{A}]_1, [\boldsymbol{d}]_1 = [\boldsymbol{t}_2\boldsymbol{A}]_1$.
The CRS consists of $\mathsf{crs} = (\mathfrak{K}, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [\boldsymbol{A}]_1 \in \mathbb{G}_1^{k\times k+1}, [\boldsymbol{U}]_2, [\boldsymbol{h}]_1 \in \mathbb{G}_1^k, [\boldsymbol{c}]_1 \in \mathbb{G}_1^k, [\boldsymbol{d}]_1 \in \mathbb{G}_1^k, \mathcal{H})$.
– **Commitment algorithm:** $\mathsf{Com}(\mathsf{crs}, M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$: to commit to message $M \in \mathbb{G}$ for party P_j, party P_i conducts the following steps:
  • It chooses random exponents $\boldsymbol{r}$ in $\mathbb{Z}_p^k$ and commits to $\boldsymbol{r}$ in $[\boldsymbol{R}]_2$ with randomness $\rho \xleftarrow{\$} \mathbb{Z}_p^{k\times k+1}$, setting $[\boldsymbol{R}]_2 = [\boldsymbol{U}\boldsymbol{\rho} + \iota_2(\boldsymbol{r})]_2 \in \mathbb{G}_2^{k\times k+1}$. It also computes a Cramer-Shoup encryption $\psi_{CS} = [\boldsymbol{C}]_1$ of $M \in \mathbb{G}_1$ under the label $\ell = P_i\|\mathsf{sid}\|\mathsf{cid}$ and the public key pk:

$$[\boldsymbol{C}]_1 = [\boldsymbol{A}\boldsymbol{r}||\boldsymbol{h}\boldsymbol{r} + M||(\boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell||\boldsymbol{C}_1||\boldsymbol{C}_2||\boldsymbol{R}))\boldsymbol{r}]_1 = [\boldsymbol{C}_1||\boldsymbol{C}_2||\boldsymbol{C}_3]_1$$

For simplicity we write $\ell' = \ell||[\boldsymbol{C}_1]_1||[\boldsymbol{C}_2]_1||[\boldsymbol{R}]_2$.

- It generates a NIZK proof $D_M = [\boldsymbol{\Pi}]_1$ that ψ_{CS} is indeed a valid encryption of $M \in \mathbb{G}_1$ for the committed $\boldsymbol{r}$ in $[\boldsymbol{R}]_2$. This requires to prove that these exponents satisfy the multi-exponentiation equations:
$$[\boldsymbol{C}_1]_1 = [\boldsymbol{A}\boldsymbol{r}]_1, [\boldsymbol{C}_2 - M]_1 = [\boldsymbol{h}\boldsymbol{r}]_1, [\boldsymbol{C}_3 = (\boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell'))\boldsymbol{r}]_1$$
The associated proof is then $[\boldsymbol{\Pi}]_1 = [\boldsymbol{\rho}^\top(\boldsymbol{A}||\boldsymbol{h}||\boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell'))]_1$.
- P_i erases r after the generation of $[\boldsymbol{R}]_2$ and $[\boldsymbol{\Pi}]_1$ but retains $D_M = [\boldsymbol{\Pi}]_1$.

The commitment is $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$.

– **Verification algorithm:** $\mathsf{VerCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$: checks the consistency of the proof $\pi_{val-enc}$ with respect to $[\boldsymbol{C}]_1$ and $[\boldsymbol{R}]_2$. and ignores the opening if the verification fails.

– **Opening algorithm:** $\mathsf{OpenCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$: reveals M and $D_M = [\Pi]_1$ to P_j.

One can easily see that $[\boldsymbol{C}_3]_1$ is the projective hash computation of a 2-universal hash proof on the language "$[\boldsymbol{C}_1]_1$ in the span of $\boldsymbol{A}$", with $[\boldsymbol{C}_2]_1$ being an additional term that uses the same witness to mask the committed message, so that $[\boldsymbol{C}]_1$ is a proper generalization of the Cramer Shoup CCA-2 encryption. Details on the $k-\mathsf{MDDH}$ Groth-Sahai proofs are given in Appendix B.

It is thus easy to see that this commitment is indeed a generalization of the FLM non-interactive UC commitment with adaptive corruption under reliable erasures (in which we switched the CRS, the Cramer-Shoup encryption and the Groth Sahai proof in the $k-\mathsf{MDDH}$ setting).

4.3 A Structure-Preserving Smooth Projective Hash Function Associated with this Commitment

Structure-Preserving Smooth Projective Hash Function. We now want to supersede the verification equation of the commitment by a smooth projective hash function providing implicit decommitment, simply using the proof as a witness. We consider the language of the valid encryptions of M using a random r which is committed into $[\boldsymbol{R}]_2$:

$$\mathcal{L}_M = \{[\boldsymbol{C}]_1 \mid \exists r \exists \rho \text{ such that } [\boldsymbol{R}]_2 = [\boldsymbol{U}\boldsymbol{\rho} + \iota_2(\boldsymbol{r})]_2 \\ \text{and } [\boldsymbol{C}]_1 = [\boldsymbol{A}\boldsymbol{r}||\boldsymbol{h}\boldsymbol{r} + M||(\boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell||\boldsymbol{C}_1||\boldsymbol{C}_2||\boldsymbol{R}))\boldsymbol{r}]_1\}$$

The verifier picks a random $\mathsf{hk} = \alpha \xleftarrow{\$} \mathbb{Z}_p^{k+3\times k+1}$ and sets $\mathsf{hp} = [\alpha \odot \boldsymbol{U}]_2$. On one side, the verifier then computes:

$$\mathsf{Hash}(\mathsf{hk}, ([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)) = [\alpha \odot ((\boldsymbol{C}_1||\boldsymbol{C}_2 - M||\boldsymbol{C}_3) - (\boldsymbol{A}||\boldsymbol{h}||\boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell')) \cdot \boldsymbol{R})]_T$$

While the prover computes $\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{\Pi}) = [\boldsymbol{\Pi} \cdot \mathsf{hp}]_T$.

– *Correctness*: comes directly from the previous equations.

– *Smoothness*: on a binding CRS, $[\boldsymbol{U}]_2$'s last column is in the span of the k first (which are simply $[\boldsymbol{B}]_2$), hence as $\mathsf{hk} \in \mathbb{Z}_p^{k+1}$, the k equations given in hp are not enough to determine its value and so it is still perfectly hidden from an information theoretic point of view.

– *Pseudo-Randomness*: Under the MDDH assumption, the subset membership decision is a hard problem, as the generalized Cramer Shoup is IND-CCA-2, and $[\boldsymbol{R}]_2$ is an IND-CPA commitment to $\boldsymbol{r}$.

Theorem 9. *Under the $k-\mathsf{MDDH}$ assumption, the above SP-SPHF is strongly pseudo-random on a perfectly hiding CRS.*

For sake of compactness, the proof is postponed to Appendix D of the additional content.

Efficiency. The rough size of a projection key is $k \times (k+3)$ (number of elements in each proof times number of proofs). It should be noted, that for a $\mathsf{CS\text{-}SPHF}$ (in the case of the oblivious transfer), instead of repeating the projection key $k+3$ times (in order to verify each component of the Cramer Shoup), one can generate a value $\varepsilon \xleftarrow{\$} \mathbb{Z}_p$, an hp for a single equation, and say that for the other component, one simply uses $\mathsf{hp}^{\varepsilon^i}$, as the trick explained in [ABB+13].

5 Application: Nearly Optimal Size 1-out-of-m Oblivious Transfer

Our oblivious transfer scheme builds upon that presented by Abdalla *et al.* at Asiacrypt 2013 [ABB+13]. In their scheme, the authors use a SPHF-friendly commitment (which is a notion stronger than a UC commitment) along with its associated SPHF in a now classical way to implicitly open the commitment. They claim that the commitment presented in [FLM11] cannot be used in such an application, since it is not "robust", which is a security notion meaning that one cannot produce a commitment and a label that extracts to x' (possibly $x' = \perp$) such that there exists a valid opening data to a different input x, even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom). Indeed, because of the perfectly-hiding setting of Groth-Sahai proofs, for any ciphertext C and for any message x, there exists a proof Π that makes the verification of C on x. However, we show in this section that in spite of this result, such a commitment can indeed be used in a relatively close construction of oblivious transfer scheme. To this aim, we use our construction of structure-preserving SPHF on FLM's commitment, simply using the decommitment value (a Groth-Sahai proof) as the witness, presented in Section 4.3.

It should be noted that the commitment used in [ACP09, ABB+13] has the major drawback of leaking the bit-length of the committed message. While in application to Oblivious Transfer this is not a major problem, for PAKE this is a way more sensitive issue, as we show in the next section. Moreover, using FLM's commitment is conceptually simpler, since the equivocation only needs to modify the witness, allowing the user to compute honestly its message in the commitment phase, whereas in the original commitments, a specific flow had to be sent during the commitment phase (with a different computation and more witnesses for the SPHF, than in the honest computation of the commitment).

5.1 A Universally Composable Oblivious Transfer with Adaptive Security Based on MDDH

We denote by DB the database of the server containing $t = 2^m$ lines, and j the line requested by the user in an oblivious way. We assume the existence of

a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and a IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter. The commitment used is the variant of [FLM11] described above. It is denoted as Com^{ℓ} in the description of the scheme, with ℓ being a label. Note that sid denotes the session identifier, ssid the subsession identifier and cid the commitment identifier and that the combination $(\mathsf{sid}, \mathsf{cid})$ is globally unique, as in [HMQ04, FLM11].

We present our construction, in Figure 3, following the global framework presented in [ABB+13], for an easier efficiency comparison (we achieve nearly optimality in the sense that it is linear in the number of lines of the database, but with a constant equal to 1 only).

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^{\kappa}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\kappa})$.

Pre-flow:
1. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen
2. Server sends pk to User

Index query on j:
1. User chooses a random value J, computes $S \leftarrow F(J)$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$
2. User computes $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1) \xleftarrow{\$} \mathsf{Com}^{\ell}(\mathsf{crs}, j, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
3. User stores $[\boldsymbol{\Pi}]_1$ and completely erases J and the random coins used by Com and $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $[\boldsymbol{C}]_1, [\boldsymbol{R}]_2$ and c to Server

Database input $(n_1, \ldots, n_t)$**:**
1. Server decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $S \leftarrow F(J)$
2. For $s = 1, \ldots, t$: Server computes $\mathsf{hk}_s \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_s)$, $\mathsf{hp}_s \leftarrow \mathsf{ProjKG}(\mathsf{hk}_s, \mathcal{L}_s)$, $K_s \leftarrow \mathsf{Hash}(\mathsf{hk}_s, (\mathcal{L}_s, (\ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2)))$, and $N_s \leftarrow S \oplus K_s \oplus n_s$
3. Server erases everything except $(\mathsf{hp}_s, N_s)_{s=1,\ldots,t}$ and sends them over a secure channel

Data recovery:
Upon receiving $(\mathsf{hp}_s, N_s)_{s=1,\ldots,t}$, User computes
$K_j \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathcal{L}_j, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}]_1)$ and gets $n_j \leftarrow S \oplus K_j \oplus N_j$.

Fig. 3. UC-Secure 1-out-of-t OT from an SPHF-Friendly Commitment (for Adaptive Security)

Theorem 10. *The oblivious transfer scheme described in Figure 3 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*

The proof is given in Appendix E of the additional content for completeness.

6 Application: Adaptive and Length-Independent One-Round PAKE

Password-authenticated key exchange (PAKE) protocols allow two players to agree on a shared high entropy secret key, that depends on their own passwords only. Katz and Vaikuntanathan recently came up with the first concrete one-round PAKE protocols [KV09], where the two players just have to send simultaneous flows to each other. Following their idea, [BBC+13b] proposed a round-optimal PAKE protocol UC secure against passive corruptions. On the other hand, [ACP09] proposed the first protocol UC secure against adaptive corruptions, and [ABB+13] built upon both [KV09] and [ACP09], to propose the first one-round protocol UC secure against adaptive corruptions. Unfortunately, both of them share a drawback, which is that they use a commitment growing linearly with the length of a password. Besides being an efficiency problem, it is over all a security issue in the UC framework. Indeed, the simulator somehow has to "guess" the length of the password of the player it simulates, otherwise it is unable to equivocate the commitment (since the commitment reveals the length of the password it commits to). Since such a guess is impossible, the apparently only solution to get rid of this limitation seems to give the users an upper-bound on the length of their passwords and to ask them to compute commitments of this length, which leads to costly computations.

In this section, we are now going to present a constant-size, round-optimal, PAKE UC secure against adaptive corruptions. It builds upon the protocol proposed in [ABB+13], using the same techniques as in the former section to avoid the apparent impossibility to use FLM's commitment.

CRS: $\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}})$.

Protocol execution by P_i with pw_i:

1. P_i generates $\mathsf{hk}_i \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_{\mathsf{pw}_i})$, $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathcal{L}_{\mathsf{pw}_i})$ and erases any random coins used for the generation
2. P_i computes $([\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2, [\boldsymbol{\Pi}_i]_1) = \mathsf{Com}^{\ell_i}(\mathsf{crs}, \mathsf{pw}_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell_i = (\mathsf{sid}, P_i, P_j, \mathsf{hp}_i)$
3. P_i stores $[\boldsymbol{\Pi}_i]_1$, completely erases random coins used by Com and sends $\mathsf{hp}_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2$ to P_j

Key computation: Upon receiving $\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2$ from P_j

1. P_i computes $H'_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathcal{L}_{\mathsf{pw}_i}, \ell_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2), [\Pi_i]_1))$ and $H_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, (\mathcal{L}_{\mathsf{pw}_i}, \ell_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2))$ with $\ell_j = (\mathsf{sid}, P_j, P_i, \mathsf{hp}_j)$
2. P_i computes $\mathsf{sk}_i = H'_i \cdot H_j$ and erases everything else, except pw_i.

Fig. 4. UC-Secure PAKE from the revisited FLM Commitment

It should be noted that we need the classical requirement for extraction capabilities (see for example [Lin11, BCPV13] for a detailed explanation), *i.e.* a password pw is assumed to be a bit-string of length bounded by $\log p - 2$, and

then one can use a bijective embedding function G mapping $\{0,1\}^{|p|-2}$ in $\mathbb{G}_1$. For the sake of simplicity, we continue to write pw_i in the high level description, but it should be interpreted as a commitment to $G(\mathsf{pw}_i)$.

The language $\mathcal{L}_{\mathsf{pw}_i}$ is then the language of valid Cramer Shoup encryptions of the embedded password $G(\mathsf{pw}_i)$, consistent with the randomness committed in the second part, and the rest of the label.

Theorem 11. *The Password Authenticated Key Exchange scheme described in Figure 4 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*

The proof is given in Appendix F of the additional content for completeness.

7 Application: Anonymous Credential-Based Message Transmission

Anonymous Credential protocols [Cha86, Dam90, CL01] allow to combine security and privacy. Typical credential use involves three main parties. Users need to interact with some authorities to obtain their credentials (assumed to be a set of attributes validated / signed), and then prove to a server that a subpart of their attributes verifies an expect policy.

In this section, we give another go to Anonymous Credential, this time to allow message recovery. This is is between Anonymous Credential but also Conditional Oblivious Transfer [Rab81] and Oblivious Signature-Based Envelope [LDB03].

We present a constant-size, round-optimal protocol that allow to use a Credential to retrieve a message without revealing the Anonymous Credentials in a UC secure way, by simply building on the commitment proposed earlier in the paper.

7.1 Anonymous Credential System

In a Attribute-Based Credential system, we assume that different organization issue credentials to users. A user i possesses a set of credential Cred_i of the form $\{\mathsf{Cred}_{i,j}, \mathsf{vk}_j\}$ where organization j assesses that the user verifies some property. (The DMV will assess that the user is indeed capable of driving, the university that she has a bachelor in Computer Science, while Squirrel Airways that she reached the gold membership, all those authorities don't communicate with each other).

A Server might have an access Policy P requiring some elements (For example being a female, with a bachelor, and capable of driving).

- $\mathsf{Setup}(1^{\mathfrak{K}})$: A probabilistic algorithm that gets a security parameter $\mathfrak{K}$, an upper bound t for the size of attribute sets and returns the public parameters param

– $\mathsf{OKeyGen}(\mathsf{param})$: Generates a pair of signing keys $\mathsf{sk}_j, \mathsf{vk}_j$ for each organization.
– $\mathsf{UKeyGen}(\mathsf{param})$: Generates a pair of keys $\mathsf{sk}_i, \mathsf{vk}_i$ for each use.
– $\mathsf{CredObtain}(\langle U_i, \mathsf{sk}_i\rangle, \langle O_j, \mathsf{sk}_j\rangle)$ Interactive process that allows a user i to obtain some credentials from organization j by providing his public key vk_j and a proof that it belongs to him.
– $\mathsf{CredUse}(\langle U_i, \mathsf{Cred}_i, \mathsf{sk}_i\rangle, \langle S, P, M\rangle)$ Interactive process that allows a user i to access a message guarded by the server S under some policy P by using the already obtained credentials.

An attribute-based anonymous credential system is called secure if it is correct, unforgeable and anonymous.

7.2 Construction

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^\kappa), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^\kappa)$.

Pre-flow:
1. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen
2. Server sends pk to User

Credential Use by user i:
1. User chooses a random value J, computes $S \leftarrow F(J)$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$
2. User computes $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1) \xleftarrow{\$} \mathsf{Com}^\ell(\mathsf{crs}, \mathsf{Cred}_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
3. User stores $[\boldsymbol{\Pi}]_1$ and completely erases J and the random coins used by Com and $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $[\boldsymbol{C}]_1, [\boldsymbol{R}]_2$ and c to Server

Database input M with policy P:
1. Server decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $S \leftarrow F(J)$
2. Server computes $\mathsf{hk}_P \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_P)$, $\mathsf{hp}_P \leftarrow \mathsf{ProjKG}(\mathsf{hk}_P, \mathcal{L}_P)$, $K_P \leftarrow \mathsf{Hash}(\mathsf{hk}_P, (\mathcal{L}_P, (\ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2)))$, and $N_P \leftarrow S \oplus K_P \oplus M$
3. Server erases everything except (hp_P, N_P) and sends them over a secure channel

Data recovery:
Upon receiving (hp_P, N_P), User computes
$K \leftarrow \mathsf{ProjHash}(\mathsf{hp}_P, (\mathcal{L}_P, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}]_1)$ and gets $M \leftarrow S \oplus K \oplus N_P$.

Fig. 5. UC-Secure Anonymous Credential from an SPSPHF-Friendly Commitment (for Adaptive Security)

Smooth Projective Hash Functions have been shown to handle complex languages [ACP09, BBC+13a], those properties can naturally be extended to Structure Preserving Smooth Projective Hash Function, allowing credentials to be expressive as disjunction / conjunction of sets of credentials, range proofs, or

even composition (having a credential from authority A signed by authority B for example).

What is really new with the Structure Preserving part is that now a user can request to have a credential on a witness by requiring a Structure-Preserving signature on it, while before scalars either required to give too much information to the server B or prevented chaining as most signatures requires some sort of Hashing (BLS requires an explicit Hash, while signature *à la* Waters requires to handle a bit per bit version of the message hindering drastically the efficiency of the protocol). This allows more possibilities in both the Credential Generation step and the policy required for accessing messages, while maintaining an efficient construction.

Theorem 12. *The Anonymous Credential Protocol described in Figure 5 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*

The ideal functionality and a sketch of the proof are given in Appendix G of the additional content for completeness.

References

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, December 2013.

[ACD+12] Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 4–24. Springer, December 2012.

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org/2005/385`.

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, August 2009.

[ADK+13] Masayuki Abe, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Tagged one-time signatures: Tight security and optimal tag size. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 312–331. Springer, February / March 2013.

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, August 2010.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666. Springer, August 2011.

[AGOT14a] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Structure-preserving signatures from type II pairings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 390–407. Springer, August 2014.

[AGOT14b] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Unified, minimal and selectively randomizable structure-preserving signatures. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 688–712. Springer, February 2014.

[BBC+13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, February / March 2013.

[BBC+13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, August 2013.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, August 2004.

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of uc-secure oblivious transfer. Cryptology ePrint Archive, Report 2015/560, 2015.

[BCL+05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 361–377. Springer, August 2005.

[BCPV13] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Analysis and improvement of Lindell's UC-secure commitment schemes. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 534–551. Springer, June 2013.

[BCV15] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Non-interactive zero-knowledge proofs of non-membership. Cryptology ePrint Archive, Report 2015/072, 2015. `http://eprint.iacr.org/`.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, May 2000.

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 94–111. Springer, March 2012.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, August 2001.

[Cha86] David Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In Franz Pichler, editor, *EUROCRYPT'85*, volume 219 of *LNCS*, pages 241–244. Springer, April 1986.

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, May 2005.

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, April / May 2002.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, February / March 2013.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, May 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, August 1998.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, April / May 2002.

[Dam90] Ivan Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 328–335. Springer, August 1990.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, August 2013.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, August 1984.

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 468–485. Springer, December 2011.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, April 2008.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 111–129. Springer, August 2007.

[HMQ04] Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 58–76. Springer, February 2004.

[JR14] Charanjit S. Jutla and Arnab Roy. Dual-system simulation-soundness with applications to uc-pake and more. Cryptology ePrint Archive, Report 2014/805, 2014.

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer, May 2005.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer, May 2001.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, December 2009.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, March 2011.

[LDB03] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 182–189. ACM, July 2003.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 446–466. Springer, May 2011.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, August 2008.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, August 1992.

A Commitments and Smooth Projective Hash Functions

A.1 Encryption

An encryption scheme $\mathcal{C}$ is described through four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$):

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of keys, a (public) encryption key pk and a (private) decryption key dk;
- $\mathsf{Encrypt}(\mathsf{ek}, M; \rho)$ outputs a ciphertext $\mathcal{C}$, on M, under the encryption key pk, with the randomness ρ;
- $\mathsf{Decrypt}(\mathsf{dk}, \mathcal{C})$ outputs the plaintext M, encrypted in the ciphertext $\mathcal{C}$ or $\perp$.

Such encryption scheme is required to have the following security properties:

- *Correctness*: For every pair of keys $(\mathsf{ek}, \mathsf{dk})$ generated by KeyGen, every messages M, and every random ρ, we should have $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, M; \rho)) = M$.
- *Indistinguishability under Adaptive Chosen Ciphertext Attack* IND-CCA ([NY90, RS92]):
  - IND-CCA: An adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two original plaintexts, and ask several decryption of ciphertexts different from challenge one.
    The $\mathsf{ODecrypt}$ oracle outputs the decryption of c under the challenge decryption key dk. The input queries (c) are added to the list $\mathcal{CT}$ of decrypted ciphertexts.

$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(M_0, M_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{pk}, \mathsf{ODecrypt}(\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, M_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot))$
6. `IF` $(c^*) \in \mathcal{CT}$ `RETURN` 0
7. `ELSE RETURN` b'

A.2 Commitments

A commitment scheme is said *equivocable* if it has a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and two algorithms

- $\mathsf{SimCom}^{\ell}(\tau)$ takes as input the trapdoor τ and a label ℓ and outputs a pair (C, eqk), where C is a commitment and eqk an equivocation key;
- $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ takes as input a commitment C, a label ℓ, a message x, an equivocation key eqk, and outputs an opening data δ for C and ℓ on x.

such as the following properties are satisfied: *trapdoor correctness* (all simulated commitments can be opened on any message), *setup indistinguishability* (one

cannot distinguish the CRS ρ generated by $\mathsf{SetupCom}$ from the one generated by $\mathsf{SetupComT}$) and *simulation indistinguishability* (one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to fake commitments), denoting by SCom the algorithm that takes as input the trapdoor τ, a label ℓ and a message x and which outputs $(C, \delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$, computed as $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$.

A commitment scheme $\mathcal{C}$ is said *extractable* if it has a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and a new algorithm

- $\mathsf{ExtCom}^{\ell}(\tau, C)$ which takes as input the trapdoor τ, a commitment C, and a label ℓ, and outputs the committed message x, or $\perp$ if the commitment is invalid.

such as the following properties are satisfied: *trapdoor correctness* (all commitments honestly generated can be correctly extracted: for all ℓ, x, if $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$ then $\mathsf{ExtCom}^{\ell}(C, \tau) = x$), *setup indistinguishability* (as above) and *binding extractability* (one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data to an input x while the commitment does not extract to x).

A commitment scheme is said *extractable and equivocable* if the indistinguishable setup algorithm outputs a common trapdoor that allows both equivocability and extractability, and the following properties are satisfied: *strong simulation indistinguishability* (one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom)) and *strong binding extractability* (one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data (not given by SCom) to an input x while the commitment does not extract to x, even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom)).

A.3 Smooth Projective Hash Functions Used With Commitments

The strong pseudo-randomness property, from [BBC+13b], is defined by the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand}}(\mathfrak{K})$ depicted in Figure 6. It is a strong version of the pseudo-randomness where the adversary is also given the hash value of a commitment of its choice (obviously not generated by SCom or SimCom though, hence the test with Λ which also contains (C, ℓ, x)). This property only makes sense when the projection key does not depend on the word C to be hashed. It thus applies to KV-SPHF, and CS-SPHF only.

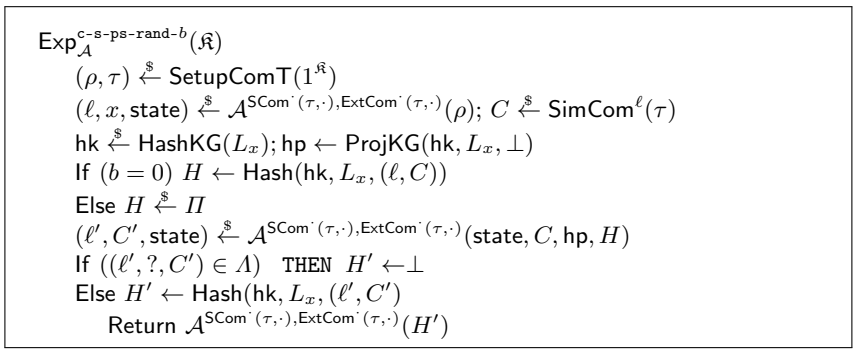

Fig. 6. Strong Pseudo-Randomness

Additional Content

B Groth Sahai Methodology

Groth and Sahai [GS08] have introduced a methodology to build Non-Interactive ZK / Witness Indistinguishable proofs of *satisfiability of pairing-product like equations*. The three types of equations handled by such proofs are the following:

A *pairing-product equation* over variables $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_m) \in \mathbb{G}_1^m$ and $\boldsymbol{\mathcal{Y}} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \in \mathbb{G}_2^n$ is of the form

$$\langle \boldsymbol{\mathcal{A}}, \underline{\boldsymbol{\mathcal{Y}}} \rangle \cdot \langle \underline{\boldsymbol{\mathcal{X}}}, \boldsymbol{\mathcal{B}} \rangle \cdot \langle \underline{\boldsymbol{\mathcal{X}}}, \Gamma \underline{\boldsymbol{\mathcal{Y}}} \rangle = t_T \ , \tag{1}$$

defined by constants $\boldsymbol{\mathcal{A}} \in \mathbb{G}_1^n$, $\boldsymbol{\mathcal{B}} \in \mathbb{G}_2^m$, $\Gamma = (\gamma_{i,j})_{\substack{1 \le i \le m \\ 1 \le j \le n}} \in \mathbb{Z}_p^{m \times n}$ and $t_T \in \mathbb{G}_T$.

A *multi-scalar multiplication equation* over variables $\boldsymbol{y} \in \mathbb{Z}_p^n$ and $\boldsymbol{\mathcal{X}} \in \mathbb{G}_1^m$ is of the form

$$\langle \underline{\boldsymbol{y}}, \boldsymbol{\mathcal{A}} \rangle \cdot \langle \boldsymbol{b}, \underline{\boldsymbol{\mathcal{X}}} \rangle \cdot \langle \underline{\boldsymbol{y}}, \Gamma \underline{\boldsymbol{\mathcal{X}}} \rangle = T, \tag{2}$$

defined by the constants $\boldsymbol{\mathcal{A}} \in \mathbb{G}_1^n$, $\boldsymbol{b} \in \mathbb{Z}_p^m$, $\Gamma \in \mathbb{Z}_p^{m \times n}$ and $T \in \mathbb{G}_1$.

A multi-scalar multiplication equation in group $\mathbb{G}_2$ is defined analogously.

A *quadratic equation in* $\mathbb{Z}_p$ over variables $\boldsymbol{x} \in \mathbb{Z}_p^m$ and $\boldsymbol{y} \in \mathbb{Z}_p^n$ is of the form

$$\langle \boldsymbol{a}, \underline{\boldsymbol{y}} \rangle + \langle \underline{\boldsymbol{x}}, \boldsymbol{b} \rangle + \langle \underline{\boldsymbol{x}}, \Gamma \underline{\boldsymbol{y}} \rangle = t, \tag{3}$$

defined by the constants $\boldsymbol{a} \in \mathbb{Z}_p^n$, $\boldsymbol{b} \in \mathbb{Z}_p^m$, $\Gamma \in \mathbb{Z}_p^{m \times n}$ and $t \in \mathbb{Z}_p$.

Groth and Sahai have detailed generic construction of the proofs π and specific instantiations under different security assumptions. We will focus on Linear Equations in the following, as they are those needed in the rest of the paper, that is to say equations with variables in only one of the two groups.

B.1 SXDH Instantiation

In order to generate a proof of such relations, the methodology invites us to commit to the witness vectors $\boldsymbol{\mathcal{X}}$ with randomness $\boldsymbol{R}$, and to $\boldsymbol{\mathcal{Y}}$ with $\boldsymbol{S}$ with two double ElGamal commitments scheme, one in $\mathbb{G}_1$ and one in $\mathbb{G}_2$ with respective commitment keys $\mathbf{u} \in \mathbb{G}_1^{2\times 2}$ and $\mathbf{v} \in \mathbb{G}_2^{2\times 2}$. As both need to be semantically secure, we will work under the SXDH assumption.

We will note $\iota_1(g_1){=}(1_1, g_1), \iota_2(g_2){=}(1_2, g_2), \iota_T(t_T) := \begin{pmatrix} 1_T & 1_T \\ 1_T & t_T \end{pmatrix}$ and focus on product pairing equations.

Assuming elements were committed in $\mathbb{G}_2$ following the way explained in 2, following [GS08] notations, we then have access to algorithms:

- $\mathsf{Prove}((\mathcal{Y}_j), (\mathbf{u}, \mathbf{v}), E; (S_j), T \in \mathbb{Z}_p^{2\times 2})$ outputs a proof π, together with $\boldsymbol{d} \in \mathbb{G}_2^{2\times n}$ commitments to the witnesses with randomness $\boldsymbol{S} \in \mathbb{Z}^{2\times n}$. The proof is composed of two elements in $\mathbb{G}_1$: $\pi = \boldsymbol{S}^\top \boldsymbol{\mathcal{A}}$
- $\mathsf{Verify}(\boldsymbol{d}, \mathsf{ck}, E, \pi)$ checks if: $\big(\iota_1(\boldsymbol{\mathcal{A}}) \bullet \boldsymbol{d}\big) = \iota_T(t_T) \odot \big(\iota_1(\pi) \bullet \mathbf{v}\big)$

There is also an algorithm that allows anyone to randomize the proof, even without the knowledge of the witnesses.

The *Soundness* and the *Witness Indistinguishability* of such a proof directly come from the security of the commitment, and the extra randomness T.

Intuitively the proof is here to compensate some part introduced by the randoms in the verification equation: $\boldsymbol{S}^\top \boldsymbol{\mathcal{A}}$ will cancel the randomness in $\big(\iota_1(\boldsymbol{\mathcal{A}}) \bullet \boldsymbol{d}\big)$.

B.2 $k-$MDDH Instantiation

[EHK+13] defined a generalization of the Groth and Sahai framework for Zero-Knowledge proof to fit with the Matrix Assumption.

Given an underlying matrix $[\boldsymbol{B}]_s$, one generate a binding CRS, by computing an additional vector in the span of $[\boldsymbol{B}]_s$:$[\boldsymbol{Bw}]_s$, and a hiding one, if there is some perturbation to this column. We name the commitment matrix $[\boldsymbol{U}]$ which is the concatenation of the matrix $\boldsymbol{B}$ and this extra column.

We define the $k+1$ vector: $\iota_s(\mathcal{X}) := (1, \ldots, 1, \mathcal{X})$, and the $k+1$ by $k+1$ matrix $\iota_T(t_T) := \begin{pmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & 1 \\ 1 & 1 & t_T \end{pmatrix}$. This allows us to extend the previous commitment to their equivalent under $k-\mathsf{MDDH}$.

To commit to an element M, one computes $[\boldsymbol{R}]_s$ with randomness $\rho \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, setting $[\boldsymbol{R}]_2 = [\boldsymbol{U\rho} + \iota_s(M)]_s \in \mathbb{G}_2^{k+1}$.

As before, there exist $\mathsf{Prove}, \mathsf{Verify}$ algorithms allowing to generate a compatible, randomizable, Zero-Knowledge proof that the committed value fulfills some pairing equation.

C Ideal Functionalities

C.1 UC Framework

The goal of the UC framework is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. It is a simulation-based model, relying on the indistinguishability between the real world and the ideal world. In the ideal world, the security is provided by an ideal functionality $\mathcal{F}$, capturing all the properties required for the protocol and all the means of the adversary. In order to prove that a protocol Π emulates $\mathcal{F}$, one has to construct, for any polynomial adversary $\mathcal{A}$ (which controls the communication between the players), a simulator $\mathcal{S}$ such that no polynomial environment $\mathcal{Z}$ (the distinguisher) can distinguish between the real world (with the real players interacting with themselves and $\mathcal{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathcal{S}$ and $\mathcal{F}$) with a significant advantage. The adversary can be either *adaptive*, *i.e.* allowed to corrupt users whenever it likes to, or *static*, *i.e.* required to choose which users to corrupt prior to the execution of the session sid of the protocol. After corrupting a player, $\mathcal{A}$ has complete access to the internal state and private values of the player, takes its entire control, and plays on its behalf.

C.2 UC-Secure Oblivious Transfer

The ideal functionality of an Oblivious Transfer (OT) protocol is depicted in Figure 7. It is inspired from [CKWZ13].

The functionality $\mathcal{F}_{(1,k)\text{-OT}}$ is parameterized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- **Upon receiving an input (Send, sid, ssid, P_i, P_j, $(m_1,\ldots,m_k)$) from party P_i**, with $m_i \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1,\ldots,m_k))$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.
- **Upon receiving an input (Receive, sid, ssid, P_i, P_j, s) from party P_j,** with $s \in \{1,\ldots,k\}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$, and reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Receive-message with the same ssid from P_j.
- **Upon receiving a message (Sent, sid, ssid, P_i, P_j) from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1,\ldots,m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to P_i and ignore further Sent-message with the same ssid from the adversary.
- **Upon receiving a message (Received, sid, ssid, P_i, P_j) from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1,\ldots,m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, m_s)$ to P_j and ignore further Received-message with the same ssid from the adversary.

Fig. 7. Ideal Functionality for 1-out-of-k Oblivious Transfer $\mathcal{F}_{(1,k)\text{-OT}}$

C.3 UC-Secure Password-Authenticated Key Exchange

We present the PAKE ideal functionality $\mathcal{F}_{pwKE}$ on Figure 8. It was described in [CHK+05].

The main idea behind this functionality is as follows: If neither party is corrupted and the adversary does not attempt any password guess, then the two players both end up with either the same uniformly-distributed session key if the passwords are the same, or uniformly-distributed independent session keys if the passwords are distinct. In addition, the adversary does not know whether this is a success or not. However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as `compromised`), the adversary is granted the right to fully determine its session key. There is in fact nothing lost by allowing it to determine the key. In case of wrong guess (the session is then marked as `interrupted`), the two players are given independently-chosen random keys. A session that is nor `compromised` nor `interrupted` is called `fresh`, which is its initial status.

Finally notice that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the password, are related passwords, are used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

The functionality $\mathcal{F}_{\text{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1, \ldots, P_n$ via the following queries:

- **Upon receiving a query (`NewSession`, sid, ssid, P_i, P_j, pw) from party P_i:** Send (`NewSession`, sid, ssid, P_i, P_j) to $\mathcal{S}$. If this is the first `NewSession` query, or if this is the second `NewSession` query and there is a record (sid, ssid, P_j, P_i, pw′), then record (sid, ssid, P_i, P_j, pw) and mark this record `fresh`.
- **Upon receiving a query (`TestPwd`, sid, ssid, P_i, pw′) from the adversary S:** If there is a record of the form (P_i, P_j, pw) which is `fresh`, then do: If $pw = pw'$, mark the record `compromised` and reply to $\mathcal{S}$ with "correct guess". If pw $\neq$ pw′, mark the record `interrupted` and reply with "wrong guess".
- **Upon receiving a query (`NewKey`, sid, ssid, P_i, sk) from the adversary S:** If there is a record of the form (sid, ssid, P_i, P_j, pw), and this is the first `NewKey` query for P_i, then:
  - If this record is `compromised`, or either P_i or P_j is corrupted, then output (sid, ssid, sk) to player P_i.
  - If this record is `fresh`, and there is a record (P_j, P_i, pw′) with pw′ = pw, and a key sk′ was sent to P_j, and (P_j, P_i, pw) was `fresh` at the time, then output (sid, ssid, sk′) to P_i.
  - In any other case, pick a new random key sk′ of length $\mathfrak{K}$ and send (sid, ssid, sk') to P_i.

  Either way, mark the record (sid, ssid, P_i, P_j, pw) as `completed`.

Fig. 8. Ideal Functionality for PAKE $\mathcal{F}_{\text{pwKE}}$

In case of corruption, the adversary learns the password of the corrupted player, after the `NewKey`-query, it additionally learns the session key.

D Proof of the Strong Pseudo-Randomness

Proof. To prove the strong pseudo-randomness, we use the following sequence of games:

Game G_0: This game is the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-0}}$.

Game G_1: In this game, we replace $(\boldsymbol{C}, \boldsymbol{R}, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ by $\boldsymbol{C}, \boldsymbol{R}, \xleftarrow{\$} \mathsf{Com}^{\ell}(\boldsymbol{M''})$ for some arbitrary $\boldsymbol{M''} \neq \boldsymbol{M}$. This game is indistinguishable thanks to strong simulation indistinguishability (Commitments and fake commitments are the same, only their decommitment witnesses differs but they are indistinguishably distributed).

Game G_2: In this game, we replace the CRS, by a perfectly binding one. (There are no more equivocation used, only real commitments). Under $k-\mathsf{MDDH}$, this is indistinguishable from the previous one.

Game G_3: In this game, before computing H', we compute $\boldsymbol{M'} \leftarrow \mathsf{ExtCom}^{\ell'}(\tau, C')$ and we abort if the decryption is not unique. In other words, if C' is not perfectly binding, we abort. However as we now have a perfectly hiding CRS, this never happen. This game is indistinguishable from the previous one.

Game G_4: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H' by a random value.

This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, the fact that $\boldsymbol{M'} \neq \boldsymbol{M}$ and C' is perfectly binding (otherwise, we would have aborted), so that $(\ell', \boldsymbol{C'}, \boldsymbol{R'}) \notin L_{\boldsymbol{M}}$, and thanks to the fact that H could have been computed as follows: $\Pi \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, \boldsymbol{C}, \boldsymbol{R}, \boldsymbol{M})$ and $H \leftarrow \mathsf{ProjHash}(\mathsf{hp}, L_{\boldsymbol{M}}, (\ell, \boldsymbol{C}, \boldsymbol{R}), \Pi)$.

Game $\boldsymbol{G_5}$: In this game, when $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H by a random value.
This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, and the fact that C is a real commitment of $\boldsymbol{M''} \neq \boldsymbol{M}$ and so that $(\ell, \boldsymbol{C}, \boldsymbol{R}) \notin L_{\boldsymbol{M}}$.
Notice that we could not have done this if $\boldsymbol{M'} = \boldsymbol{M}$, since, in this case, we still need to use hk to compute the hash value H' of C'. We are handling this (tricky) case in the following game.

Game $\boldsymbol{G_6}$: In this game, we replace H by a random value, in the case $\boldsymbol{M'} = \boldsymbol{M}$. So now H will be completely random, in all cases (since it was already the case when $\boldsymbol{M'} \neq \boldsymbol{M}$).
Finally, we write $[\boldsymbol{r'}]_2$, the vector extracted from $\boldsymbol{R'}$. There are two cases:

1. If $[\boldsymbol{C'}_1]_T = [\boldsymbol{A} \cdot \boldsymbol{r'}]_T$ In this case, since C' extracts to $\boldsymbol{M}$, this means that $(\ell', \boldsymbol{C'}, \boldsymbol{R'}) \in L_{\boldsymbol{M}}$, and its hash value H' could be computed knowing only hp and r'. Therefore, the hash value H of C looks random by smoothness.
2. Else $\boldsymbol{C'}_1$ is not in the correct span, the rows of the matrix Γ in Equation and the two vectors $\Theta(\boldsymbol{C})$ and $\Theta(\boldsymbol{C'})$ are linearly independent. Then, even given access to the hash value H' of $\boldsymbol{C'}$ and the projection key hp, the hash value H of $\boldsymbol{C}$ looks perfectly random.

The following games are just undoing the modifications we have done, but keeping H picked at random

Game $\boldsymbol{G_7}$: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we compute H' as originally (as the hash value of C').
This game is indistinguishable from the previous one thanks to the smoothness of the SPHF.

Game $\boldsymbol{G_8}$: In this game, we do not extract $\boldsymbol{M'}$ from C' nor abort when C' is not perfectly binding.

Game $\boldsymbol{G_9}$: In this game, we now move back to the perfectly hiding CRS. Under $k - \mathsf{MDDH}$ this game is indistinguishable from the previous one.

Game $\boldsymbol{G_{10}}$: In this game, we now compute C as originally using SimCom. This game is indistinguishable thanks to strong simulation indistinguishability.
We remark that this game is exactly the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-1}}$.

□

E Proof of the Oblivious Transfer Scheme

To prove theorem 10, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$.

Compared to the protocol presented in [ABB+13], the difficulty here arises from the fact that the commitment from [FLM11] needs the CRS to be hiding in order to be equivocable and that such a CRS forbids the use of smoothness for the SPHF. Instead of being able to replace all the commitment queries by simulated (fake) commitments from the beginning of the proof, one has to only allow the extraction trapdoors (for the Cramer-Shoup encryption inside the commitment, and for the CPA encryption of J) and to deal carefully with the properties of the SPHF before being able, at the end of the proof, to turn the CRS into a hiding one and simulate the commitments. More details follow (the description of the simulator can be found in the last game).

Game $\boldsymbol{G}_0$: This is the real game.

Game $\boldsymbol{G}_1$: In this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs $(n_1, \ldots, n_t)$ and j sent by the environment to the server and the user. In all the subsequent games, the players use the label $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game $\boldsymbol{G}_2$: In this game, we just replace the setup algorithms so that the simulator knows the trapdoor for extracting both the Cramer-Shoup encryption, and the $\mathsf{Encrypt}_{\mathrm{cpa}}$ encryption. Note that we do not change anything more in the setup, implying the CRS remains a CRS for a perfectly-sound Groth-Sahai setting. Corruptions are handled the same way.

Game $\boldsymbol{G}_3$: We first deal with **honest servers**: when receiving a commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$, the simulator extracts the committed value j from $[\boldsymbol{C}]_1$. Instead of computing the key K_s, for $s = 1, \ldots, t$ with the hash function, it chooses $K_s \xleftarrow{\$} G$ for $s \neq j$.

Since $[\boldsymbol{C}]_1$ is extracted to j, then, with an hybrid proof applying the smoothness for every honest server, on every index $s \neq j$, the hash value K_s is indistinguishable from a random value for $s \neq j$.

In case of corruption, everything has been erased. This game is thus indistinguishable from the previous one under the smoothness of the smooth projective hash function.

Game $\boldsymbol{G}_4$: We continue to deal with **honest servers**: when receiving a commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$, the simulator extracts the committed value j from $[\boldsymbol{C}]_1$. Instead of proceeding as the server would do on $(n_1, \ldots, n_t)$, the simulator proceeds on $(n'_1, \ldots, n'_t)$, with $n'_j = n_j$, but $n'_s = 0$ for all $s \neq j$. Since the masks K_t, for $t \neq s$, are random from the previous game, this game is perfectly indistinguishable from the previous one.

Game $\boldsymbol{G}_5$: We continue to deal with **honest servers** and more precisely with the computation of S. If the user and the server are still honest until the server has received an honestly-generated $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, c)$ from the honest user, the simulator does not extract J and compute $S \leftarrow F(J)$, but directly sets $S \leftarrow (J')$, with J' a random value, for both players.

With an hybrid proof, applying the $\mathsf{IND\text{-}CPA}$ property for each session, one can show the indistinguishability of this game with the previous one.

Game G_6: We continue to deal with **honest servers** and more precisely with the case described in the previous game. In this case, the simulator now directly sets S as a random value, instead of setting $S \leftarrow F(J')$.
With an hybrid proof, applying the PRF property for each session, one can show the indistinguishability of this game with the previous one.

Game G_7: We continue to deal with **honest servers** and now describe how the simulator generates K_j, in case the user and the server are still honest until the server has received an honestly-generated $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, c)$ from the honest user. In this case, thanks to the additional random mask S, the simulator can send a random N_j on behalf of the server, and postpone the computation of K_j at the time the user actually receives this value.
Since the adversary does not know any decommitment information $[\boldsymbol{\Pi}]_1$ for the commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$ (thanks to the $\mathsf{IND\text{-}CCA}$ properties of the Cramer-Shoup and Groth-Sahai encryptions), this hash value K_j is indistinguishable from a random value, applying the pseudo-randomness for every honest server. If the server involved in the pseudo-randomness gets corrupted, we are out of this case and can thus abort it.
In case of corruption of the server, everything has been erased. In case of corruption of the user, the simulator receives the good value n_j and is able to choose R (which is a random value unknown to the adversary, and because all the other K_t are independent random values too) such that

$$R \oplus \mathsf{ProjHash}(\mathsf{hp}_j, (L_j, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2)) \oplus N_j = n_j.$$

This game is thus indistinguishable from the previous one under the pseudo-randomness.

Game G_8: We continue to deal with the same case. On behalf of an honest server, the simulator proceeds with the database $(n'_1, \ldots, n'_s)$, with $n'_s = 0$ for all s even $s = j$. Since the masks $K_s \oplus S$, for any $s = 1, \ldots, t$, are all independent random values (this comes from the fact that the K_t, for $t \neq s$, are independent random values, and S is independently random), this game is perfectly indistinguishable from the previous one.
We remark that it is therefore no more necessary to know the index j given by the ideal functionality to the honest user in order to correctly simulate the server. But note that the knowledge of this index is still necessary to simulate the user (in particular in case of corruption). We show in the next games how to get rid of this knowledge.

Game G_9: In this game, we completely replace the setup algorithms so that not only the simulator knows the trapdoor for extracting both the Cramer-Shoup encryption, and the $\mathsf{Encrypt}_{\mathrm{cpa}}$ encryption, but the CRS also becomes a CRS for a witness indistinguishable Groth-Sahai setting. This game is indistinguishable from the former one under the k-MDDH assumption.

Game G_{10}: In this game, we deal with **honest users**, assuming that the simulator still knows the index j which has to be committed to by the honest users and still computes the commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$ honestly. However, in case of corruption, it computes the proof $[\boldsymbol{\Pi}]_1$ using the simulation trapdoor

rather than the real witnesses. This game is indistinguishable from the former one since, in the witness-indistinguishable setting, simulated proofs are distributed as real proofs.

Game $\boldsymbol{G}_{11}$: In this game, we still deal with **honest users**, by without using anymore the knowledge of j when simulating. On behalf of an honest user, the simulator chooses an index j' at random, and computes honestly the commitments $([\boldsymbol{C}]_1, [R_2])$ to this value j'. In case of corruption, it computes the proof $[\boldsymbol{\Pi}]_1$ corresponding to the real j it has just learnt using the simulation trapdoor as in the former game. Since the proofs are already simulated, this is indistinguishable from the former game thanks to the semantic security of the Cramer-Shoup encryption. The argument uses an hybrid proof and is the same as [CF01, Theorem 8] or [FLM11, Theorem 1].

When it finally receives the values (hp_s, N_s) from the adversarial server, the simulator computes, for $s = 1, \ldots, t$, $[\boldsymbol{\Pi}_s]_1$ corresponding to a commitment of s, $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, (L_s, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}_s]_1)$ and gets $n_s \leftarrow S \oplus K_s \oplus N_s$, giving it the database submitted by the server.

The only problem would arise if an adversarial user committed to j and was able to open its commitment to j' by computing the correct $[\boldsymbol{\Pi}_{j'}]_1$. But since the commitment is universally composable, there is a neglibible probability for it to be able to compute this witness. Thus, the security again relies on the pseudo-randomness, making this game indistinguishable from the previous one.

Game $\boldsymbol{G}_{12}$: We can now make use of the functionality, which leads to the following simulator:

- when receiving a Send-message from the ideal functionality, which means that an honest server has sent a pre-flow, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as pre-flow;
- after receiving a pre-flow pk (from an honest or a corrupted server) and a $\mathtt{Receive}$-message from the ideal functionality, which means that an honest user has sent an index query, the simulator generates $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1) \xleftarrow{\$} \mathsf{Com}(\mathsf{crs}, j', \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ for a random index j' and $c \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{pk}, S)$, for a random value S, and sends $[\boldsymbol{C}]_1$, $[\boldsymbol{R}]_2$ and c during the index query phase on behalf of the honest user;
- when receiving a commitment $[\boldsymbol{C}]_1$, $[\boldsymbol{R}]_2$ and a ciphertext c, generated by the adversary (from a corrupted user), the simulator extracts the committed value j, and uses it to send a $\mathtt{Receive}$-message to the ideal functionality. It also decrypts the ciphertext c as J, and computes $S = F(J)$;
- when receiving $(\mathsf{hp}_1, N_1, \ldots, \mathsf{hp}_t, N_t)$ from the adversary (a corrupted server), the simulator computes, for $i = 1, \ldots, t$, $[\boldsymbol{\Pi}_s]_1$ corresponding to a commitment of s, $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, (L_s, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}_s]_1)$ and gets $n_s \leftarrow S \oplus K_s \oplus N_s$, giving it the database submitted by the server. It uses these values to send a Send-message to the ideal functionality.
- when receiving a $\mathtt{Received}$-message from the ideal functionality, together with n_j, on behalf of a corrupted user, from the extracted j, instead of

proceeding as the server would do on $(n_1, \ldots, n_t)$, the simulator proceeds on $(n'_1, \ldots, n'_t)$, with $n'_j = n_j$, but $n'_s = 0$ for all $s \neq t$;
- when receiving a commitment $[\boldsymbol{C}]_1$, $[\boldsymbol{R}]_2$ and a ciphertext c, generated by an honest user (*i.e.*, by the simulator itself), the simulator proceeds as above on $(n'_1, \ldots, n'_t)$, with $n'_s = 0$ for all s, but it chooses S uniformly at random instead of choosing it as $S = F(J)$; in case of corruption afterwards, the simulator will adapt S such that $S \oplus \mathsf{ProjHash}(\mathsf{hp}_j, \ell, (L_j, [C]_1, [R]_2), [\Pi]_1) \oplus N_j = n_j$, where n_j is the message actually received by the user.

Any corruption either reveals j earlier, which allows a correct simulation of the user, or reveals $(n_1, \ldots, n_t)$ earlier, which allows a correct simulation of the server. When the server has sent his flow, he has already erased all his random coins.

However, there would have been an issue when the user is corrupted after the server has sent is flow, but before the user receives it, since he has kept Π_1: this would enable the adversary to recover n_j from N_j and hp_j. This is the goal of the ephemeral mask S that provides a secure channel.

F Proof of the PAKE Scheme

To prove theorem 11, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}_{\text{PAKE}}$ and the adversary $\mathcal{A}$. For simplicity, in the proof we are going to use pw loosely to designate an encoding $G(\mathsf{pw})$. A decryption can lead to two cases, either a valid encoding $G(\mathsf{pw})$ which can then be reverted to a password pw, or an invalid one. In this last case, the simulator assumes the password pw to be $\perp$.

For the sake of simplicity, since the protocol is fully symmetric in P_i and P_j, we describe the simulation for player P_i in order to simplify the notations.

We say that a flow is *oracle-generated* if the tuple $(\mathsf{hp}_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2)$ was sent by an honest player P_i (or the simulator) and received without any alteration by the adversary. It is said *non-oracle-generated* otherwise.

Game $\boldsymbol{G_0}$: This is the **real game.**

Game $\boldsymbol{G_1}$: First, in this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs pw_i and pw_j sent by the environment to the players. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

In the following, Step 1. is always generated honestly by the simulator, since the hashing and projection keys do not depend on any private value.

Game $\boldsymbol{G_2}$: In this game, we just replace the setup algorithms so that the simulator knows the trapdoor for extracting the Cramer-Shoup encryption. Note that we do not change anything more in the setup, implying the CRS

remains a CRS for a perfectly-sound Groth-Sahai setting. Corruptions are handled the same way.

Game $\boldsymbol{G}_3$: In this game, we deal with the case where P_i **receives a flow oracle-generated from P_j, and they have identical passwords.** When P_i receives an oracle-generated flow from P_j, the simulator checks whether the two passwords sent by the environment for P_i and P_j are identical. If so, $\mathcal{S}$ computes both hash values using Hash and not $\mathsf{ProjHash}$. More precisely, it computes $H'_i = \mathsf{Hash}(\mathsf{hk}_j, (\mathcal{L}_{\mathsf{pw}_j}, \ell_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2)))$ (with $\ell_i = (\mathsf{sid}, P_i, P_j, \mathsf{hp}_i)$). If the passwords are distinct, it does not change anything. Recall that it is able to do so since it generated the hashing keys on their behalf.

Thanks to the correctness of the SPHF, this game is indistinguishable from the former one.

Game $\boldsymbol{G}_4$: In the next two games, we deal with the case where P_i **receives a flow oracle-generated from P_j, but P_j has been corrupted, and they have distinct passwords**. In this case, $\mathcal{S}$ has received the password pw_j of P_j at the corruption time of P_j (pw_j was anyway already known), and knows the corresponding opening data $[\Pi_j]_1$, which it computed honestly on behalf of P_j (since it still knows pw_j). If this password is the same, it does not change anything. If the passwords are distinct, then $\mathcal{S}$ computes H'_i as before, but chooses H_j at random: this means that we replace $\mathsf{Hash}(\mathsf{hk}_i, (\mathcal{L}_{\mathsf{pw}_i}, \ell_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2))$ by a random value, while $[\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2$ have been simulated by Com with an opening value $[\Pi_j]_1$ for $\mathsf{pw}_j \neq \mathsf{pw}_i$.

Using an hybrid proof, this game is indistinguishable from the former one using the smoothness of the SPHF.

Game $\boldsymbol{G}_5$: We conclude for this case: if the passwords are distinct, P_i chooses a random key.

Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_6$: In the next two games, we deal with the case where P_i **receives a flow oracle-generated from P_j, and P_j is still honest, and they have distinct passwords.** The simulator checks whether the two passwords sent by the environment for P_i and P_j are distinct. If so, $\mathcal{S}$ replaces $\mathsf{Hash}(\mathsf{hk}_j, (\mathcal{L}_{\mathsf{pw}_j}, \ell_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2))$ by a random value, the first time it is computed (and uses the same random value the second time it is computed by the partner, and thus only if this is the same password).

In case the player on which we made the modification is later corrupted, this is out of this case, and thus we abort this hybrid game and go to the next one. Using an hybrid proof, this game is indistinguishable from the former one using the pseudo-randomness.

Game $\boldsymbol{G}_7$: We conclude for this case: $\mathcal{S}$ sends a random key to P_i.

Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_8$: In the next two games, we deal with the case where P_i **receives a non-oracle-generated flow** $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$**.** Since this pair is fresh, either $[\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2$ is new or hp_j (and thus the label) is new. Since $[\boldsymbol{R}_j]_2$ is

used in the label of the Cramer-Shoup encryption $[\boldsymbol{C}_j]_1$, either both of them are new or not. In both cases, $\mathcal{S}$ can extract the committed value pw'_j on behalf of P_j.

If this password is the same than that of P_i (which the simulator can easily check, still having access to the private values sent by the environment), $\mathcal{S}$ still computes both H_j and H'_i as before.

Otherwise (or if the extraction fails), the $\mathcal{S}$ computes H'_i as before, but chooses H_j at random:

Under the smoothness, with an hybrid proof, one can show the indistinguishability of the two games.

Game $\boldsymbol{G}_9$: Finally, when P_i receives a non-oracle-generated flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$ that extracts to a different password than that of P_i (or for which extraction fails), then $\mathcal{S}$ sets the session key of P_i as random.

Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_{10}$: In this game, we completely replace the setup algorithms so that not only the simulator knows the trapdoor for extracting the Cramer-Shoup encryption, but the CRS also becomes a CRS for a witness indistinguishable Groth-Sahai setting. This game is indistinguishable from the former one under the k-MDDH assumption.

Game $\boldsymbol{G}_{11}$: In this game, we still use the knowledge of pw_i to compute $[\boldsymbol{C}]_1, [\boldsymbol{R}]_2$ but in case of corruption, it computes the proof $[\boldsymbol{\Pi}]_1$ using the simulation trapdoor rather than the real witnesses. This game is indistinguishable from the former one since, in the witness-indistinguishable setting, simulated proofs are distributed as real proofs.

Game $\boldsymbol{G}_{12}$: We do not use anymore the knowledge of pw_i when simulating an honest player P_i. On behalf of an honest user, the simulator chooses a password pw'_i at random, and computes honestly the commitments $([\boldsymbol{C}]_1, [R_2])$ to this value pw'_i. In case of corruption, it computes the proof $[\boldsymbol{\Pi}]_1$ corresponding to the real pw_i it has just learnt using the simulation trapdoor as in the former game. Since the proofs are already simulated, this is indistinguishable from the former game thanks to the semantic security of the Cramer-Shoup encryption. The argument uses an hybrid proof and is the same as [CF01, Theorem 8] or [FLM11, Theorem 1].

The only problem would arise if an adversarial user committed to pw'_i and was able to open its commitment to pw''_i by computing the correct $[\boldsymbol{\Pi}]_1$. But since the commitment is universally composable, there is a negligible probability for it to be able to compute this witness. Thus, the security again relies on the pseudo-randomness, making this game indistinguishable from the previous one.

The private values of P_i are thus not used anymore in Step 1. and Step 2. The simulator only needs them to choose how to set the session key of the players. In the ideal game, this will be replaced by a NewKey-query that will automatically deal with equality or difference of the passwords, or `TestPwd`-query for non-oracle-generated-flows.

Game $\boldsymbol{G}_{13}$: This is the ideal game. Now, the simulator does not know the private values of the honest players anymore, but can make use of the ideal functionality. We showed in Game $\boldsymbol{G}_{12}$ that the knowledge of the private values is not needed anymore by the simulator, provided it can ask queries to the ideal functionality:

Initialization: When initialized with security parameter $\mathfrak{K}$, the simulator first runs the commitment setup algorithm obtaining a witness indistinguishable crs in which it knows the trapdoors for the extraction of the Cramer-Shoup encryption and the simulation of the Groth-Sahai proofs. It initializes the real-world adversary $\mathcal{A}$, giving it crs as common reference string.

Session Initialization: When receiving a message $(\texttt{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ from $\mathcal{F}_{pwKE}$, $\mathcal{S}$ executes the protocol on behalf of P_i as follows:

1. $\mathcal{S}$ generates honestly $\mathsf{hk}_i \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L})$ and $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathcal{L})$;
2. $\mathcal{S}$ computes $([\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2, [\boldsymbol{\Pi}_i]_1)\mathsf{Com}^{\ell_i}(\mathsf{crs}, \mathsf{pw}_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell_i = (\mathsf{sid}, P_i, P_j, \mathsf{hp}_i)$;
3. $\mathcal{S}$ sends $\mathsf{hp}_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2$ to P_j.

If P_i gets corrupted, $\mathcal{S}$ recovers the password pw_i and computes the corresponding proof $[\Pi_i]_1$, which it is able to give to the adversary.

Key Computation: When receiving a flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$:

- if the flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$ is non-oracle-generated, $\mathcal{S}$ extracts the password pw'_j (or sets it as a dummy value in case of failure of extraction). $\mathcal{S}$ then asks for a $\texttt{TestPwd}$-query to the functionality to check whether pw'_j is the password of P_i. If this password is correct, $\mathcal{S}$ sets $\mathsf{pw}_i = \mathsf{pw}'_j$, computes the corresponding proof $[\Pi_i]_1$, as well as H_j and H'_i, and then sk_i, that is passed to the NewKey-query ($\texttt{compromised}$ case). If the password is incorrect, $\mathcal{S}$ asks the NewKey-query with a random key ($\texttt{interrupted}$ case).
- if the flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$ is oracle-generated but the associated P_j has been corrupted, then $\mathcal{S}$ has recovered its password pw_j and has been able to compute the corresponding proof $[\Pi_j]_1$. It can thus compute sk_j, that is passed to the NewKey-query ($\texttt{corrupted}$ case).
- if the flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$ is oracle-generated and the associated P_j is still uncorrupted, $\mathcal{S}$ asks the NewKey-query with a random key (normal case).

One can remark that the NewKey-queries will send back the same kinds of session keys to the environment as in Game $\boldsymbol{G}_{12}$: if a player is corrupted, the really computed key is sent back, in case of impersonation attempt, the $\texttt{TestPwd}$-query will address the appropriate situation (correct or incorrect guess), and if the two players are honest, the NewKey-query also addresses the appropriate situation (same or different passwords).

G Sketch of Proof for Anonymous Credential-Based Message Transmission

G.1 Ideal Functionality

The ideal functionality for Anonymous Credential-Based Message Transmission is given in Figure 9. The server S agrees to send a message M to the user, as soon as his credentials Cred comply with the policy P.

The functionality $\mathcal{F}_{\mathrm{AC}}$ is parametrized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_N$ via the following queries:

- **Upon receiving an input $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, M, P)$ from party P_i**, with $M \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, M, P)$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, P)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{Cred})$ from party P_j**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, M, P)$ is not recorded. Otherwise, reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$ and send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, M')$ to P_j where $M' = M$ if the credentials comply with the policy P, and $M' = \perp$ otherwise. Ignore further $\mathtt{Receive}$-message with the same ssid from P_j.

Fig. 9. Ideal Functionality for Anonymous Credential-Based Message Transmission $\mathcal{F}_{\mathrm{AC}}$

G.2 Idea of the Proof

We sketch the proof by giving the simulator $\mathcal{S}$ such that no polynomial environment $\mathcal{Z}$ can distinguish between the real world (with the real players interacting with themselves and $\mathcal{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathcal{S}$ and $\mathcal{F}$) with a significant advantage. Recall that the adversary is adaptive, which means that it can corrupt any player at any time during the execution of the protocol.

The main idea is that we use an extractable and equivocable commitment, which allows the simulator (while simulating the user) to be able to open it to any credential. This will be useful in case of adaptive corruptions. Indeed, in this case, if the credentials were correct, the simulator can adapt them and the randomness so that they seem to belong to the adequate language. From the server's side, the extractability of the commitment enables the simulator to know whether it has to send the correct message (obtained from the functionality in case the user is corrupted) or not.

This leads to the following simulator:

- when the simulator receives a Send-message from the ideal functionality, it knows than an honest sender has sent a pre-flow. It thus generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as a pre-flow.

– when the simulator receives a `Receive`-message from the ideal functionality, it knows than an honest user has sent a pre-flow. It also has received a pre-flow pk (from an honest or a corrupted sender). It then generates an equivocable commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1) \xleftarrow{\$} \mathsf{Com}^{\ell}(\mathsf{crs}, \mathsf{Cred}_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ and a ciphertext $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$ where S is a random value.
– when it simulates an honest server who receives these values $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1)$ and c from a corrupted user, it decrypts the ciphertext c as J, and computes $S = F(J)$. Next, it extracts the committed values (Cred_i), which it uses to send a `Receive`-message to the ideal functionality.
– when it simulates an honest user receiving (hp_P, N_P) from a corrupted server, it computes $K \leftarrow \mathsf{ProjHash}(\mathsf{hp}_P, (\mathcal{L}_P, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}]_1)$ and gets $M \leftarrow S \oplus K \oplus N_P$. It uses this value in a Send query to the ideal functionality.
– when it simulates an honest server and receives a `Received`-message from the ideal functionality, giving it M sent to the corrupted user, it proceeds with this value M.
– when it simulates an honest server facing an honest user (the simulator itself), on the behalf of which it generated the commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1)$ and the ciphertext c, it uses $M = 0$ and chooses S at random (instead of computing it honestly with $F(J)$). This value S will be adapted during the simulation in case of corruption afterwards (which gives it the message M received by the user in case his credentials comply with the policy), so that $M = S \oplus K \oplus N_P$.